An Informing Service Based on Models Defined by Its Clients

Juan Ricardo Bauer Mengelberg
Colegio de Postgraduados, Montecillo, Edo. De Mexico, Mexico

jbauer@colpos.mx

Abstract

The developments of the transdiscipline Informing Science not only furnish useful criteria for those creating or offering informing services, but also result in a new responsibility associated with them. Checking the information quality features of the processes involved provides many suggestions to improve the resulting service. The paper describes an informing service for agribusinesses that will offer information about the impact of changes in real world data on their businesses, rather than just report these changes. A customer formulates a mathematical model which includes real world variables – representing external data relevant to his business – as well as his own variables. The service will implement ways to obtain up-to-date values for the variables offered to its customers, typically through intelligent agents or information brokers. The impact of changes in values of the external variables on other variables results from an execution of the model. The client may query his data, but also can define situations in which the he wishes to be warned, with some degree of urgency, that something happened which merits an action or decision on his part. These alarms are delivered through an email or a text message. The components of the service are described. Different attributes of the information's quality were examined, and changes in the design to comply with them improved the informing service.

Keywords: informing service, agribusiness, communication, actionably timely available, interpretable, mathematical models.

Introduction

Services that offer up to date information to their customers are blossoming, and their coverage is ever more extensive. In many parts of the world, it is now possible to be informed almost immediately that something has happened that either affects you or interests you in other ways. Cellular phone networks have added another dimension to the already impressive advances the Internet contributed to these possibilities of knowing what is happening. It is also true that not all information received will be useful or even interesting. Thus, ample opportunities arise in the field of Informing Science (IS) to improve the informing processes. The fields that comprise IS can achieve this, if the focus is to "*provide* their *clientele* with *information* in a *form, format,* and *schedule* that maximizes its effectiveness" (Cohen, 1999). This paper describes the informing service called FLAG, which is an acronym for Cash Flow in Agribusinesses, since it was designed specifically for that sector in Mexico, though it may just as well serve entrepreneurs of other business sectors and in any other country. Its fea-

tures were designed so that the resulting service would have almost every one of the quality of information attributes cited in the Informing Science literature.

In a previous paper, Cohen (1997) had remarked that the definition of IS points to three interrelated components: the client (who has a task to perform that requires information for its completion), the delivery system (for providing information), and the informing environment that creates information to aid the clients complete their tasks.

FLAG initially addressed the four primary attributes of information suggested by Gackowski (2005). "It must be interpretable, of significant impact, timely available and credible. In particular, timely access to information that significantly impacts the client's business, plays a fundamental role, since it may constitute a significant competitive advantage". Specifically, in the context of a business environment, Gackowski adds the all important cost elements to these attributes, formulating them in terms of the information's use in decision making or carrying out specific actions that take advantage of the information obtained. "The information must be interpretable at a non-excessive cost, should be ready on time, must be credible, which implies that it must be objective, unbiased, accurate, correct, accurate and current, and must be presented in a way suitable for the receiver, so that it can be interpreted easily and without error". However, the concept of "actionably timely available", formulated in that paper, is probably the most important attribute of the information a businessman needs: information that arrives too late to make a difference looses its significance. Hence, it was adopted as the guiding principle of FLAG.

Additionally, since the importance of cost considerations cannot be overstated, the system would have to provide an efficient way to determine when to obtain data and which of its clients needed to be informed when changes occurred. And once again, the client will be empowered to participate actively in this determination. Birdsall (2009) formulates it concisely: "The role of the client can be considered from at least two perspectives: (1) as a participant in the use of an informing process; (2) as a participant in the process of designing and developing an informing process."

Of course some of the other attributes of an informing service are essential. The use of mathematical models addresses mainly – but not exclusively – that the information must be interpretable and of significant impact. On the other hand, the nature of the service as a centralized mechanism for many customers to use it, should provide credibility, currency, and accurateness, especially through the careful selection of its information sources, but also through the use of the appropriate tools to obtain current values of the data it furnishes to its customers.

As can be seen, some of these attributes refer to the information, not only to the service itself. Thus, the definition of information plays a significant role from a theoretical point of view. Stahl (2006) explores the meaning of misinformation, disinformation, and bias, after examining some of the most notorious definitions. He quotes Wiener (1954, p.17): "Information is a name for the content of what is exchanged with the outer world as we adjust to it, and make our adjustment felt upon it," thus emphasizing the processes involved in information sharing rather than the entity itself. Another very suitable definition of information in the context of FLAG is: "When *data* acquires context-dependent meaning and relevance, it becomes information. Furthermore, we obviously expect information to represent valid knowledge on which users can rely for rational action" (Ulrich, 2001).

After citing Cohen (1999): "A final precept of informing science is that the elements of an informing system (e.g., task, technology, structure and people) interact in a manner that is sufficiently complex such that changing the characteristics of one component (e.g., technology) can have a significant impact on the behavior of other components," Gill and Bhattacherjee (2007) state that "An important implication of this interrelatedness is that technology must play a particularly central role in informing science research, since it tends to be the element within such systems that is changing most rapidly, and is therefore the engine that drives much of the change

in system behavior." FLAG not only addresses these changes; its data models and subsystems also were designed to allow them to be updated easily, especially reacting to significant changes in the technologies involved. This is especially true in the most technology-dependent components: the processes and tools used to obtain timely information for its customers, and the communication modes which allow the prompt delivery of urgent messages to the customers.

To achieve these goals, FLAG uses mathematical models to integrate the data component (what the client knows or determines) as *his* variables, or *client variables* of the model, and the informational component, which is represented by what were called *real world* (RW) variables. The service will obtain current values of the RW variables, execute the client's model and deliver the information. Normally, the client will have access to his model anytime, but in some cases an urgent communication will be necessary, and this gives rise to the *alarms* component of FLAG, included precisely for that purpose.

After an introduction to the informing service and the data structures it uses, the type of mathematical models, the values of their variables and the relations between these are described. The model also includes specifications that will determine when the model it should be executed and an alarm generated if appropriate. Some comments about the execution and the nature of the formulae are presented, and the data structures used to store the elements of the system are described. Finally, FLAG is described in terms of IS: how does it address the quality of information attributes selected for this purpose. The paper concludes with remarks about the service, its usefulness and suggested future research. An appendix contains more detailed explanations of some of the features of the service.

FLAG – An Informing Service

Introduction

The main components of the service, and their objectives, are shown in Figure 1. Some comments should not only explain the terms and designations used, but also serve as an introduction to every one of these components. However, the main concept underlying FLAG is that a customer formulates his own mathematical model through which he will be informed of changes in data of his business environment.

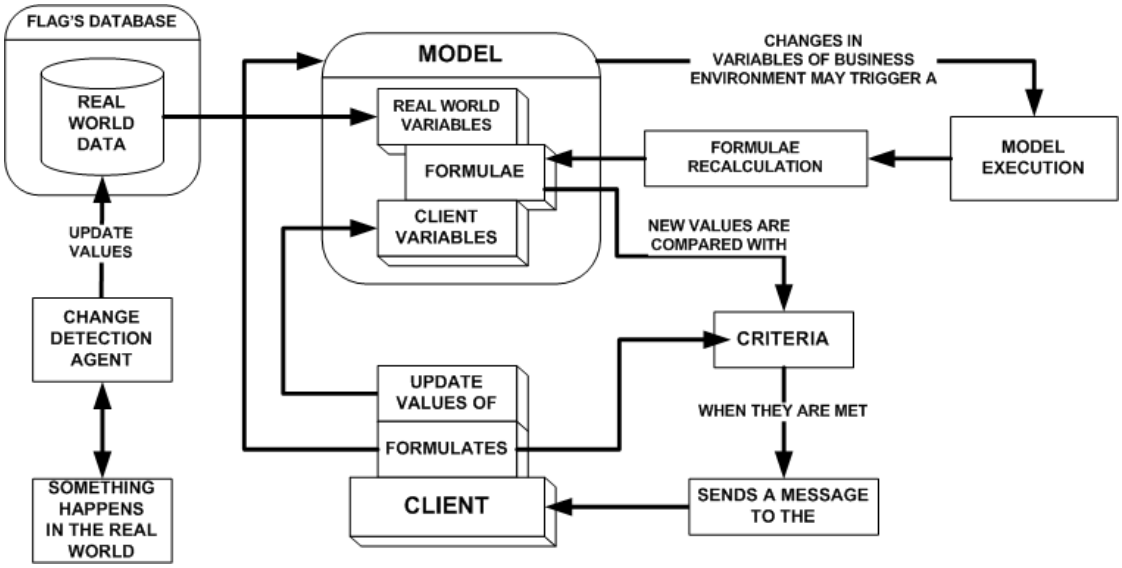


Figure 1: The main components of FLAG.

Clients provide and update the values of the *client variables*. The models of all clients share the current values of their *RW variables*, obtained and updated by the provider of the service. Whenever changes in RW data are detected, the corresponding rules defined by the clients will trigger the execution of their models, whenever necessary. As a result of an execution, alarm criteria may result in an urgent message delivered to the client.

Entities and concepts are described before the data structures, mostly database table, since many of the fields of the latter would not be clear without an introduction to their meanings.

A Client and His Model

Every client of the service has his own database, which usually will reside on the host computer used by FLAG. It contains contact information, especially his phone number and email address, some means by which he can prove his identity (authentication), his model, and the data values of his own variables (client variables). There are two ways to update any part of these items anytime: using the client's own computer, connected to the host computer or even using his own copy of the database, or by invoking the functions to do so with the programs offered as an online service by FLAG. The processes, interfaces and procedures are not described in detail in this paper, since that is not its purpose. Some of them are described in the Appendix, and others can be found on the FLAG site (http://jbauerm.com/flag/english/index.html).

The client's model serves two purposes: it reflects the processing of up-to-date data in the context of his business, and, through the use of real world variables offered by the service, indicates exactly what type of information he needs. He specifies the model's variables, which will include three types of these: client, real world and calculated variables, each of the latter provided with a formula which will result in the values of the variable. The formulae will be explained after the example of the model shown in Figure 2.

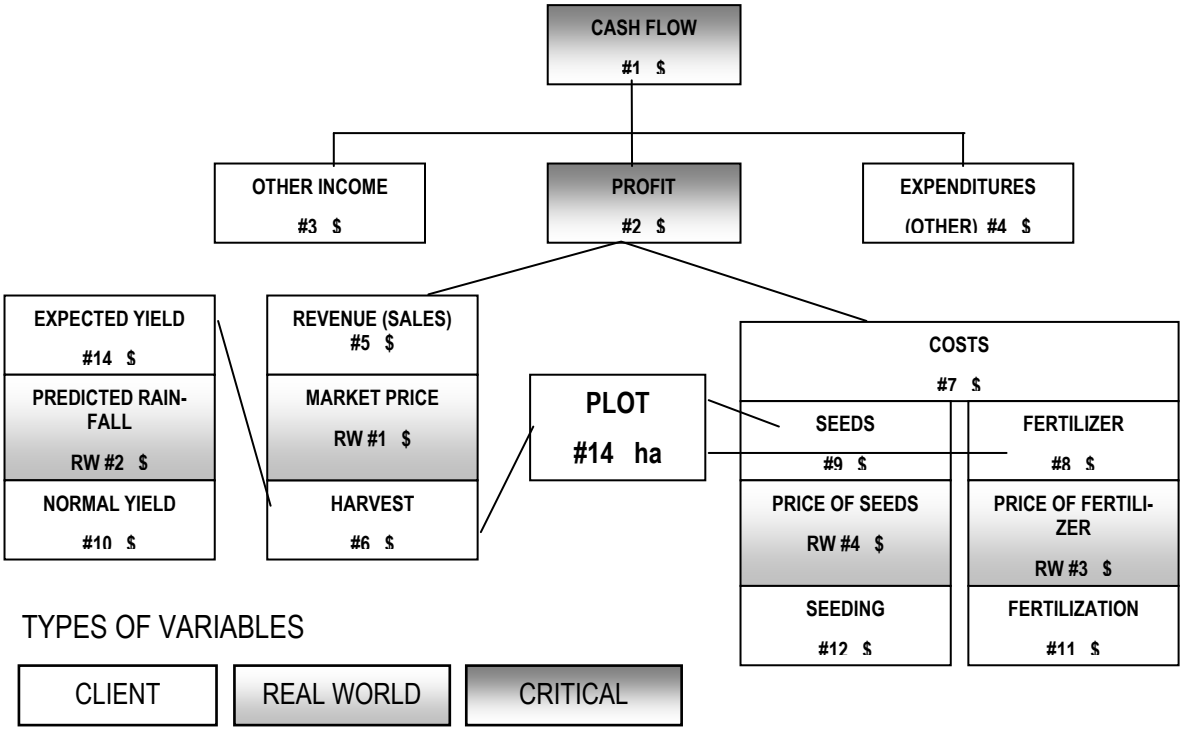


Figure 2: A simple model.

A variable is really a vector of 24 elements. Since they generally will be used as values corresponding to time periods, they are called *periods* of the variable. Thus the term applies to the value, as well as to the number of element of the vector. Though this is not a design constraint, usually all variables of a given model will use these periods in the same manner, and in our examples they are consecutive months. However, any other unit of time may be applicable: fortnights, quarters, even days are possible alternatives. Note that almost all explanations that follow will refer to the value of one period, even though they may apply to all of them. For example, when we compare the value of a variable with its previous value, the programs actually perform comparisons on several periods. The way the system will determine the range of periods to include in any particular operation or comparison is not explained in this paper, since it handles many exceptions and involves several details. For example, not all clients will use the values of a RW variable in the same positions of their vectors: the service offers June/2008 through May/2010, and the customer's model has a planning horizon comprising Jan/2009 through March/2010; not all 24 periods must be used. In this case, the client's period 1 correspond to period 8 of the RW values.

Every one of these values may in turn be a number or a random variable. As a result of decisions made during the design of the system, FLAG only admits discrete probability distributions with a maximum of 4 values. All elements of the same variable must be of the same type - either numbers or random variables. Of course, for some of the periods of a variable with random values, one may specify a distribution with only one value: the system will handle the variable as having random values during the execution of formulae.

An important note: we decided to - incorrectly – use the term *scalars* for variables that to not have random values. The proper name would have been deterministic values, or just numbers, since the term scalar, in Mathematics, usually refers to a number, as opposed to a vector.

The formulae offered by FLAG

The decision regarding the type of formulae available to compute values of variables, based on the values of other variables, was to define them in 4 *sub-totals,* which replace the use of parentheses. Many potential users of these models might not be mathematically inclined, and formulating their operations in stages apparently simplifies matters for them. Of course the main objective is to avoid mistakes. In the Appendix there is a description of a way to test a formula, so the user can be sure he defined it correctly. These sub-totals can be thought of as temporary vectors. Should a formula need more than 4 of them, a dummy variable is created with the first 4, and the final formula will use it as its first operand.

The forms used to introduce a formula will be illustrated using the CASH FLOW variable (#1) of the example. Figure 3 shows the initial form used to introduce the corresponding formula (default values of 1 subtotal with 2 operands are offered).

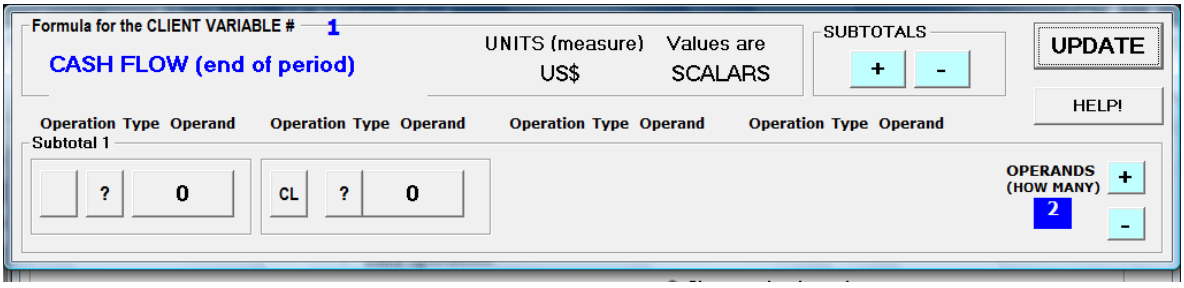


Figure 3: Form to select how many subtotals the formula will use.

A click on the "+" button of sub-totals adds an additional frame for the second sub-total , and two successive clicks on the "how many operands" correct the number of operands in the second sub-

total to 4. As illustrated in Figure 4, operations are chosen from a list of those available for that instance, which appears upon clicking on the corresponding command button. Unary operations can only be indicated for the first operand of a subtotal.

A note for experienced or very thorough readers: the "Shift" operation, which is really unary since it has only one operand, is implemented as a binary operation, where the second operator is a constant indicating the direction and amount of the shift. In the example, the formula will compute the new value of every period using value of the previous period.

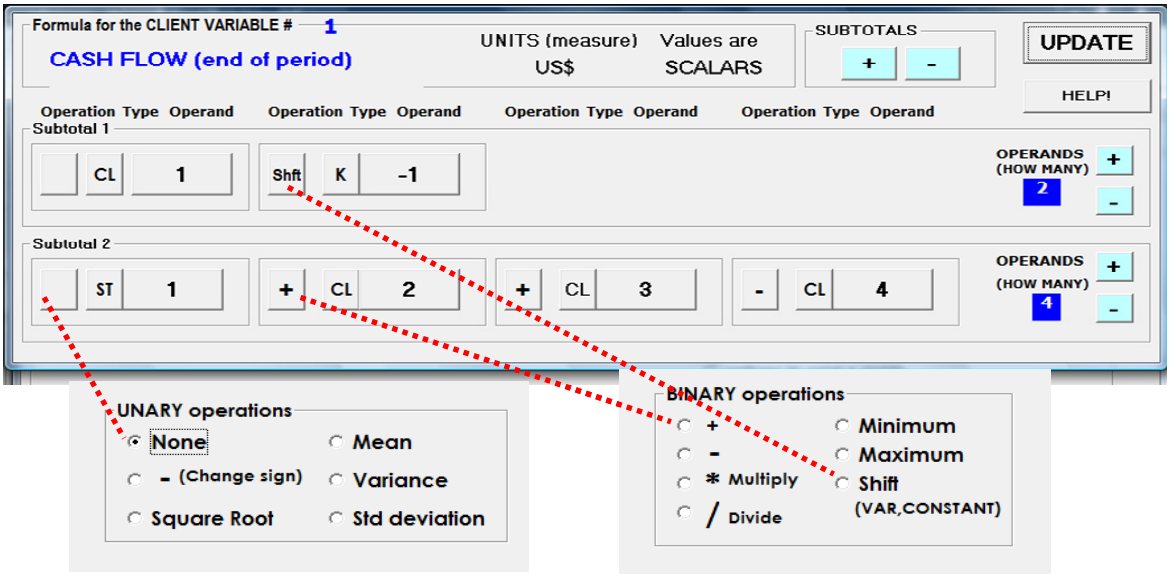


Figure 4: The formula for the variable CASH FLOW to illustrate valid operations.

A similar process is used to indicate the operands. First the type of operand must be chosen from a list of the valid types: CL, RW or K=constant. This selection will produce a display of all valid variables of that type. If a variable that is already an operand of another formula is selected, the system informs the user of that fact, and offers to create a synonym (the concept and need of such synonyms is described below). If this offer is accepted, the system will show the number of the new variable. The lists which display these items are not shown in Figure 4, since they are superimposed on the rest of the form.

Formulae with random variable operands

The mean, variance, and standard deviation operators were included for their use in formulae involving random variables: all of them result in a scalar. Arithmetic operations are performed assuming the independence of the operands and using the random-variable-arithmetic described briefly below. Note that the square root, minimum and maximum operations do not apply to random valued operands. Of course the operations are performed for every one of the periods involved, which not necessarily will be all 24.

The distribution of the sum of two random variables V1 and V2 is the convolution of their density functions. Since they are both discrete, the operation can be described as follows. Let V1 be a random variable that, for period 1, takes the values v1(k) with probabilities p1 (k).,for k = 1 to 3. Similarly, let V2 be a random variable, whose period 1 has the probability distribution v2(j) – p2 (j),for j = 1 to 2. The probability distribution of V3 = V1 + V2 is (V1(k) +Vp2 (j), p1 (k). p2 (j)) for all k = 1 to 3, j = 1 to 2. Any repeat values of V3 are combined, and the corresponding probabilities added. If more than 4 values remain, a special routine determines the best way to adjust a new probability density to either 2, 3 or 4 values. The "shape" of the distribution resulting from

the sum is reproduced as closely as possible. For example, a unimodal distribution will probably result in 3 values, whereas a bimodal one will be adjusted with 2 or 4 values. For a skew distribution, the determination of the resulting distribution is a bit awkward, but the other cases are easy to handle. However, the two main objectives usually are to preserve both the mean and standard deviation of the distribution to be replaced by reducing the domain of the variable to a maximum of 4 values.

Operations between a scalar and a random-variable are performed on every value of the random variable. The result will have the same probability distribution, exception for the domain of the variable. Using the variable V1 used in the previous example, the operation V1 * V15, where V15 is a scalar, will result in a random variable with values vi(k) * V15, k = 1 to 3, and the same probabilities p1(i) as V1. Since it is easy to be confused by the insufficient notation, in which an indication of the period was omitted to simplify the explanations, note that all of these values correspond to one period of the variables.

Synonyms of variables

In the model used as an example, it can be seen that the variable PLOT is an operand of 3 different formulae. This means that the underlying structure is not a tree, which Weiss (1993, p.87) defines recursively as: "A tree is a collection of nodes. The collection can be empty. Otherwise, it consists of a distinguished node called root, and zero or more sub-trees, each of whose nodes are connected by a directed edge to the root." This means that every node (excepting the root) is connected to exactly one parent node. Though the fact that it is not a tree poses no problem *per se*, it prevents the partial computations of large models, when only a part of the model is affected by some change in variables. There are several ways to attain a tree structure to store such a hierachical model, as Gupta & Prasad (2005) have named this situation. The way FLAG resolves this situation is through the use of *synonyms*, a designation of a variable that does not have its own values, but shares those of what we call its base variable. Thus, the variable PLOT will have two synonyms. The formula for HARVEST would use the PLOT variable, but the other two (Fertilizing and Seeds) would use its synonyms as their operands.

Level of a formula and sub-models

It is essential to avoid that the calculation of a formula precedes that of each of its operands. In FLAG, this is achieved using an algorithm to assign levels to the formulae, and during the execution of a model the formulae will be evaluated in ascending order of their levels. The algorithm starts assigning a level of zero to RW variables and client variables without a formula (not calculated variables). Then levels are assigned iteratively to every formula, where the new level is one greater than the level of its operand of highest level. Should the variable being computed be an operand of its own formula, it will not be included in this process. The process concludes when no levels are changed, or a loop is detected. In the latter case, the program that performs this function will indicate the variables involved in circular references, and the model, i.e. its formulae, must be changed to eliminate the loops.

A similar process will determine the *sub-model* of every variable of the model. This is possible precisely since the structure is now a tree. From the root of the tree representing the model, in the example Variable # 1 (CASH FLOW), the root nodes of its sub-trees are numbered consecutively as "11", "12" and "13", where these are interpreted as strings of characters, not as numbers. The same numbering is repeated for each of these nodes. In our example, only "12" – PROFIT _ has such a sub-tree, so its nodes are numbered: V#5 (REVENUE) will be "121", whereas V#7, COSTS, will be "122" This in turn allows the impact of a change to be computed individually. Suppose there is a change in the value of RW#1 (MARKET PRICE), with sub-model "1211". Only the variables whose sub-models are "121", "12", and "1" must be recomputed. The use of

sub-models in partial executions, illustrated using the cash flow example, can be found in the Appendix.

We are constantly asked why FLAG was not implemented using spreadsheets. In favor of choosing this platform, "Excel has a complex and finely tuned algorithm for choosing the fastest sequence and the minimum number of cells required to calculate the correct answer" (Excel's Smart Recalculation, 2008). This would obviously have saved a lot of work, since not only the correct order of execution of formulae would have been guaranteed, but also some performance issues could have been avoided. However, the specifications of models such as the ones used by FLAG, including 24 values for the variables, would have been very difficult. "Array formulae are one of Excel's most powerful features, although not always the easiest to use" (Array Formulae, 2008). The inclusion of random variable values, with their probability distributions, further complicates matters considerably. Since FLAG was designed for clients who do not necessarily have the skills to do this, it was well worth the effort to design interfaces that eliminated many of the difficulties and subtleties involved in preparing such a complex spreadsheet. And of course, several of its features would have been almost impossible to include without specific routines, which even in Excel are programmed in VBA (Visual Basic for Applications).

Rules that determine executions of a client's model and the resulting alarms

As a component of his model, the client indicates when it should be executed – besides the periodic frequency he indicates – and whether and how he wishes to be alerted that something happened that requires his immediate attention. These rules will allow the service to determine if a model should be executed as a consequence of changes in the values of its RW variables. During such an execution of a model, the other set of rules will indicate if the client needs to be alerted through an alarm, of a critical impact on his affairs.

To indicate a triggering device that will cause his model to be executed, he may specify a percentage or absolute range of values for any of his RW variables, so that, should the new value fall outside this range, the model will be executed. He may also accumulate several minor changes, so that his model is recomputed even if no individual RW variable changes sufficiently to trigger an execution by itself. The ways in which he specifies these rules are similar to his alarm criteria, described below. Their detailed description would only have caused confusion – precisely due to this similarity – and their comprehension is not essential to understand the rest of the system.

Generation of an alarm during the execution of a model

An alarm in FLAG has three elements: the client who will receive it, the message to be conveyed and the communication channel used to deliver it. The client will select one or more variables that he considers *critical***:** when their values change significantly, he will want to be alerted, so that either the information delivered through the alarm allows him to decide what he has to do to take advantage of the information, or induce him to query his model to know the current values of whatever indicators he included in the model. A complete description of the alarms component can be found in Bauer Mengelberg and Velazquez (in press).

The triggering device used to generate an alarm is the severity index of the execution, obtained by adding the severities assigned to significant variations in the values of the critical variables. It was introduced to enable the accumulation of several changes, not severe enough to trigger an alarm individually, but whose combined effect merits such a warning to be issued. The value of the severity index in turn will determine the need to send an alarm, and the communication channel through which it should be delivered. FLAG, in its present version, offers the 4 communication modes presented in Table 1, which also serves to show how the customer indicates his urgency.

Table 1: An example of the range of severity index and associated communication channel

Range of severity index	21-40	41-60	61-75	76 or more
Communica-tion Channel	Email: query your model	Email containing the message	SMS conveying the message	Personal phone call by employee

Note that the first mode, where his email does not contain any specific information, was included for the case that the client does not want to expose his data to a possible unauthorized reading of an email. Usually, he will either not use this method, or use only this communication mode, by assigning it to a range of 21 to 100.

When an alarm is generated by the execution of a model, a record is created in the GENERATED-ALARMS table of Flag's database. For every client, a record will contain the information of an alarm that either should be, or already has been, delivered to that client. The table, besides the information necessary to send the message (email, phone number or even address), will have fields containing the text of the message, the code indicating through what communication channel it should be delivered, and an indicator of "message delivered," which is set to "FALSE" when the record is generated. Upon delivery, the field is set to TRUE and the date and hour of completion is recorded, especially to attend to client complaints, but also to provide billing, accounting, CRM and statistical information. A process executed periodically, for example every 10 minutes, will process the records of this table and send all pending alarms.

Alarm Criteria for Critical Variables of the Model

A criterion for a particular (critical) variable consists of a critical interval for the values of one or more periods of the variable. A critical interval is one that includes the value with which the new value is to be compared: the critical event occurs whenever the new value falls *outside* that interval. To simplify the description, we will use 2 variables of the cash flow example. For each, sample criteria are presented in Table 2. The explanation of the columns of the table includes some additional concepts.

Column 2 (Critical period): he may be interested in variations of the values of the variable for more than one month, so he can furnish criteria for several periods of the same variable.

Table 2: Examples of alarm criteria for 2 critical variables

Variable	Critical period	Compare with	Critical or Emergency	Interval limits Left	Right	Interpretation	Severity
Profit	6	Previous	Critical	-20	40	Amplitude	20
	7	Previous	Emergency	-40	60	Amplitude	100
Cash flow	6	Cumulative	Critical	-5	5	Percentage	30
	6	---	Emergency	100	99999	Absolute	100
	7	----	Absolute	0	99999	Absolute	60

Column 3 (compare with): he can ask FLAG to compare the new value with the previous value. But he may also indicate that this comparison should be with the last value he knows of that period of the variable, presumably the one resulting from the execution that produced the previous

alarm. We call this the cumulative difference since the last alarm. Of course, it does not apply if no comparison is made, in the case of an absolute interval.

Column 4 (Critical or emergency): an *emergency interval* – somewhat larger than the critical interval – may be added as part of a criterion. When the emergency criterion is met, the severity will be larger.

Column 5 (Critical interval): For each one of these periods, the critical interval is specified by the values of its extreme points. They are indicated by a variation (the previous value plus or minus the numbers indicated), or, if appropriate, an absolute interval.

Column 6 (Interpretation): the numbers indicated to define the interval may be percentage variations of the previous value, amplitudes of the interval or the extreme points themselves: this is called an *absolute* interval.

Column 7 (Severity): an integer value between 1 and 100 is assigned to the range to reflect the relative importance perceived by the client of such a variation.

For example, the interpretation of the criteria for his "cash flow" variable would be as follows. He should be notified urgently whenever his cash flow falls below 100 in the current month (6). Should the cash flow vary by more then 5%, a severity of 30 will be added to the severity index. As far as next month is concerned, he requests an alarm if his predicted cash flow were negative, but with less urgency, unless his profit also will suffer significant changes.

The Data Models

Since we believe that the best way to understand a system is to study its data elements, especially the fields and what they will contain, the data models are used as a shortcut to a description of the entire system. The Appendix will focus on the functions of the system: what the users of the system can or have to do, and how they perform these activities.

Only conceptual models are presented, since the details are unnecessary to understand the service and its nature. Tables of a relational data base are used to describe the data models, even though they may not correspond to their technical implementations. Highlighted fields represent the primary indices of the tables. Though several other indices were included in the actual models, they are not shown here. Some of the fields are named and commented so that they make sense to the reader. Fields marked "NF" are not really fields, but other structures, such as a database table, a list or a plain file associated with the field; those marked as "G" indicate they are represent several fields grouped for the purpose of these explanations.

The services' data model

The central model, used by FLAG, the service provider, includes a list of RW variables, the sources of information from which their values will be obtained, and the current data values of each of them. Additionally, the criteria which will determine which models must be executed as a result of changes in the values of their variables are stored, and administrative components complete the model. Tables 3, 4, and 5 contain descriptions of the fields of the tables of the database used by the service itself, and some of the fields (or groups of them) are described. An additional table, the "generated alarms" table, is described briefly in the section dedicated to alarms.

Comments on the fields of the RW Variables table shown in Table 3 follow.

Description: contains the name of the variable, the label (called *shortie*) that will appear whenever the name is too long for the form on which the name is displayed, and a description that may provide additional information about the variable. The units of the values of the fields are also included in this group.

Table 3: The Real World Variables table

FIELD	DESCRIPTION AND COMMENTS
RW-Var-id	A unique number assigned to the variable, it is the principal index.
Description (G)	Name, label and a description, including explanation and other information.
Factor	To be applied to the values of the variable (also referred to as scale).
Periods (G)	What are periods and the date of the first period
Source for updates(NF)	Agent, broker, reports, other sources.
When-must-update (NF)	Time units and number (e.g. 4 hours) and specifications of ranges of hours or days (e.g. Monday-Friday, and/or) 8:00 – 16:00).
Cost-and-price(NF)	Costs of updating and price to be paid by clients for these updates.

Factor: an integer that specifies the values are to be multiplied by 10 to the power indicated by the factor. E.g. If value is 13 and the factor = 2, the real value is 1300. It may be negative. Necessary since values are stored as integers.

Periods: contains the units of time (months, days, weeks or others), as well as the date of the period corresponding to the first value stored for that variable.

Source information: the name of the agency that provides the values, e.g., NYSE. Also the URL, web-page or other, how the information is accessed (typically retrieval agents) and the parameters needed by the agent to extract the desired value.

When-must-update: Update frequency in time units and number (e.g., 4 hours) and specifications of ranges of hours or days (e.g. Monday-Friday, and/or 8:00 – 16:00). Also the necessary dates (last updated and next update due).

The structure used to store current values of the RW variables, shown in Table 4, is self-explanatory. Once again, there are other fields, though there use is technical, for example for security issues and to enhance performance.

Table 4: The RW Variables Value table

FIELD	DESCRIPTION AND COMMENTS
RW-Var-id	A unique number assigned to the variable. Is the primary index.
Values (NF)	24 integer values of the variable.
Last-updated	Last instance of arrival of new values.

FLAG has a table where it keeps track of all its clients, whose elements are listed in Table 5. Besides contact and contract information, the table contains the rules formulated by the client to determine when his model should be executed as a result of changes in values of the RW variables included in the model.

Table 5: FLAG-CLIENTS table

FIELD	DESCRIPTION AND COMMENTS
Client-id	A client number assigned by the service.
Name-etc (G)	Name, address, email, phone numbers
Contract-information (NF)	Start and end date of contract, billing information, fees, AR.
Model-last-executed	Date and time of last execution of his model
Uses-triggered-executions	"NO" means he did not request such executions, so his model will only be run periodically
Periodicity(NF)	His model will be executed in these time intervals. Consists of a number and the corresponding unit of time (e.g. 3 days).
Cumulative-trigger-contributions	The sum of contributions to the triggering decision of RW variables that changed sufficiently to produce an element of this decision.
Execute-model	Yes or no: should be executed in the next batch of triggered executions. The model will be executed if the cumulative trigger contributions are at least 100. Note: when the model is executed, the cumulative trigger contributions are set to 0 (zero) and the decision is set to "NO".

There is another component of the system which is not described in this paper: FLAG may offer calculated RW variables to its clients. It includes a model similar to those described for the clients, but offers additional operations to its formulae. This allows the service to combine several variables into another one, which in turn is offered to its customers. Though the main function of this model is to build random valued variables from the data it obtains from different sources, it allows the preparation of indices, trends, predictions and transformations of the basic RW variables. Executions of this model are triggered by changes in the variables' values. The fields of the table that contains the corresponding rules are shown in Table 6.

Table 6: The RW-Execution-trigger table

FIELD	DESCRIPTION AND COMMENTS
RW-Var-id	The number that identifies the variable.
Type-of-Reference-value	(Of the variable). This term is explained in the section devoted to alarms.
Triggering Interval	An interval around the reference value. If the new value falls outside this interval, the variable contributes to the decision to trigger an execution.
Interval-limits-refer-to	Percentage, Amplitude of interval or absolute values (see alarm criteria, where the corresponding terms are explained).
Contribution-to-trigger	An integer that will be added to the cumulative trigger contributions.

The Client's Data Model

Every client will have a database and other data files of his own, meaning other clients will have similar but distinct files. The entities and fields are shown as tables of a database, though, once again, this does not correspond to the actual physical implementation of the structures. Tables 7 through 11 show the fields of the database's tables. The client table shown in Table 7 consists of a unique record.

Table 7: The CLIENT table

FIELD	DESCRIPTION AND COMMENTS
Name-address (G)	Name and address, email and phone numbers for communication.
Options (NF)	FLAG offers options to its clients, which might influence the cost of the service. For example, uses-random-variables, uses-alarms, uses-triggered-executions (NO means only periodic executions).
Security- options (NF)	FLAG offers levels of confidentiality and security. Includes scrambling all data values, but mainly constrains use of the data.
Periodicity (NF)	How often his model should be executed. Corresponds to the field of the same name in the FLAGS-CLIENTS table.
Planning-horizon	The number of periods (values of variables) he will use.
What-are-periods	Meaning of the 24 values: months, quarters, days or whatever.
First-period	The (default) date of the first value of his variable vectors.
Communication-mode (NF)	Part of his specifications for alarms: the communication channel applicable to ranges of the severity index of an execution of his model.

In the Client-variable table, shown in Table 8, several fields were also grouped. The alarm-criteria, which are described elsewhere in this paper, groups the fields referring to every one of the critical variables; note that there is another database table, shown below as Table 11, that contains the intervals for every critical period of the variable.

Table 8: The CLIENT-VARIABLE table

FIELD	DESCRIPTION AND COMMENTS
Var-id	A number specified by the creator of the model
Has-synonyms	A technical field. A yes value indicates it has at least one synonym.
Description (G)	Descriptive name for the variable, a label and comments he may add.
Last-updated	Applies to the non-calculated value, those the client will update himself.
Units	How the values are interpreted (dollars, tons,…)
Factor	Explained previously regarding RW variables.
First-period	The date corresponding to the first value of this variable (overrides default).
Formula (NF)	Only for calculated variables. They are stored in files described below.
Level-etc (G)	The level of the formula and the sub-model of the variable. Includes “dad”.
Data-values (NF)	Values of the 24 periods of the variable. The structure is not shown here.
Alarm-criteria (G)	The alarm criteria for this variable.

Every model of a client will choose the RW variables he needs from those offered by the service, and indicate the frequency of updates for every one of them, as shown in Table 9, where it can be seen that several attributes of these variables are copied into this table, especially to enable executions without a connection to the service’s database.

Table 9: The RW-VARIABLES-OF-THE-CLIENT-MODEL table

FIELD	DESCRIPTION AND COMMENTS
RW-Var-number	The number assigned to this variable in this model. Note that he may not want to use the number of the variable in FLAG's table.
RW-var-id	The number of the corresponding variable in the Real World Variables Table (of FLAG).
Name	Descriptive name for the variable. This field, as all the other ones, is copied from the FLAG table so as to enable their use without access to the original table.
Units	How the values are interpreted (dollars, tons,…)
Factor	This field was explained in preceding tables.
First-period	The date corresponding to the first value of this variable.
Submodel	The sub-model of the variable.
Dad	The previous variable in the sub-model (the variable that has this RW variable as an operand of its formula).

The concept of synonyms of other variables was introduced previously. For these, only the fields shown in Table 10 are necessary, since all the other information about them is shared with their corresponding basic variable, that is, the one that contains their values.

Table 10: The SYNONYMS table

FIELD	DESCRIPTION AND COMMENTS
Synonym-id	A unique number assigned to the variable.
Basic-var-id	The number of the variable of which it is a synonym (the variable that provides the values of this one). It may be a client or and RW variable.
Submodel-and-dad	The sub-model and dad of this synonym.

For every variable designated as critical, the intervals are defined for one or more periods of that variable. Table 11 shows the main data elements of this table.

Table 11: The VARIABLE-PERIOD-ALARM table

FIELD	DESCRIPTION AND COMMENTS
Cl-Var-number	A number specified by the creator of the model.
Period	The period to which the alarm criterion applies.
Critical-interval (NF)	2 "Limits" and how they are interpreted.
Emergency-interval (NF)	2 "Limits" and how they are interpreted.
Type-of-Reference-value	The new value will be compared with: the previous value (last execution), the last known value (resulting form last execution that generated an alarm), or "absolute" (does not apply, since the interval was defined as an absolute range of values).
Critical-severity	An integer representing the severity associated to the critical interval.
Emergency-Severity	An integer representing the severity associated to the emergency interval.

The data model for the formulae

Formulae are not stored in a database table. Instead, they are included in a plain file, one formula per record, described in Figure 5. The main reason for using such files is processing efficiency, as will be pointed out in the section about this topic. Note: Actually, the alarm criteria for a variable are stored in the same record used by its formula, to avoid having to access a different structure during the execution of the model. However, the layout of the formula record presented here does not reflect this fact: its inclusion would have deterred from the objective of presenting the fields of the record.

THE_FORMULA :	here we refer to it as the formula of "THIS variable"
Computing_level	(computed) level of the formula
Factor_to_be_applied	the factor as defined for THIS variable
has_synonyms	important when only sub-models are executed)
Submodel	e.g. 1311
Dad:	variable of which THIS variable is an operand
Formula_to_dad	" ", "+", "-" (additive operand or not)
How_many_subtotals	1 to 4
A-Subtotal(1 To 4)	every subtotal consists of the fields shown below
how_many_operands	from 1 to 4
operation(1 To 4)	numeric-code "+" = 1, "-" = 2, and so on
the_operand(1 To 4)	every operand consists of 3 fields
type_of_operand	Client variable, RW, Constant, Subtotal
operand_num:	the number of the variable or the value of the constant
uses_values_of	this is the number of the "base" variable should the operand be a synonym of another variable.

Figure 5: A record to store the formula of a variable.

The values of all variables are also stored in plain files. This will allow the client to have several sets of values, for example, if he wants to use his model for simulation purposes, or wants to save generations of data values for any purpose they may serve. It also simplifies the transfers of files containing updated values form and to the client. Of course processing efficiency played a role in the decision to use such files, which on the other hand, require more programming and control activities than the use of databases.

Other Features of FLAG

The client's own information sources

A client may indicate some information sources of his own. He may do this for any reason, but some degree of confidence or reliability he assigns to a particular source, or the exclusivity he might obtain from its use, probably rank amongst these motives.

The degree of confidentiality of his model and data

FLAG includes a very basic role based access control component (Bauer Mengelberg, 2005). In the programs used by the service itself, every user of the system's programs will have a role, so that he may only perform tasks for which he was granted permission. This will limit the functions users may use in any the clients' models.

The client may choose among certain options offered by the service, to protect his data from unauthorized users. Besides being able to limit access to the basic functions (such as changing the model or the data), he can use several features to prevent strangers from seeing or using his data. For example, he can indicate that an employee of the service can execute his model, but cannot perform any query or read the data in any other way unless he provides the appropriate password. The tables used by the AC component were not shown, since this component was not included in the general description of FLAG presented in this paper.

Finally, all his data files may be scrambled or protected otherwise. This is especially true when such files are sent over the Internet: they are deformed by the sender, and recovered by the receiver. Of course all scrambling and encoding routines, which are my own and made to order for every use, such as this one in FLAG, are kept secret, as far as this is possible.

The execution of a model

Essentially, the process called execution of the model consists of computing the new values of its calculated variables, using their respective formulae. The order in which the formulae must be computed was already described before. A formula will have a level higher than any of its operands, so that, since formulae are executed in ascending order of their levels, the condition is satisfied. When the client defines or changes his model, either by adding or eliminating variables or changing their formulae, these levels are recalculated.

Whenever a critical variable is recomputed, its critical criteria are evaluated for every period. This in turn will determine if, at the end of the execution, an alarm will be generated to the client.

The client's access to his data

FLAG includes powerful querying facilities, described in the Appendix. A client may construct a query with easy to use (the term user friendly is too ambiguous). He may catalogue the queries he will use frequently, since, if his model is large, he would have to perform the selection and indicate the order in which he wants to obtain information over and over again. For example, he may want to know the values of certain variables, and order them in a way useful for a particular purpose. Or he may want to show a sub-model, or just certain types of variables. Queries include the specification of the periods he wishes to see, since presenting all 24 of them might crowd the monitor or be of no additional use. The inclusion of vertical or horizontal reports is also useful: he might want to see several variables as columns for a number of periods, or vice-versa, show several periods as columns for the variables he selects as rows of his query. He can eliminate a column or row from the result, especially if he chooses to print the display shown on his monitor. In the first version of these products, the client uses software installed in his computer. The corresponding programs which he will invoke on the Internet, to be performed in the host computer, are presently being developed and debugged.

Of course there will also be printed reports available to the client. He can ask for periodic output from his model to be sent to him, or print a report on his own computer whenever he wishes. The functions that allow him to define his reports are similar to those used to define queries. Finally, queries can also serve as an export tool: results can be exported to a spreadsheet, or to other structures, especially for the use of the data in other systems.

How to Achieve Cost Effectiveness

Since FLAG is supposed to provide its clients competitive advantages, the costs involved are of utmost importance. Though the pricing and administrative components are still in the preliminary design phase, an outline of their elements will be presented, since without these, the rest is meaningless. Both the service provider and the client must do their part to achieve cost effectiveness.

The provider will attempt to reduce his costs in several ways, but so will the client. Three of the cost elements will be discussed here. For each, we point out how they are addressed and the tasks to be performed by both parties of the service.

Cost of obtaining current information

The cost of acquiring data must be kept as low as possible, by the proper selection of information sources, but also avoiding excessive updates. Thus, the system must include mechanisms that only invoke retrieval agents or other data-gathering devices whenever necessary, that is, if any of its customers need an update at that moment.

Of course, marketing plays a role in these costs as well. If a data item is expensive, no matter why, the service should attempt to have as many clients as possible that will use that particular item in their models.

Systematizing all the activities necessary to obtain the data values whenever needed, will contribute to reduction in human resources and control activities, and the inclusion of the data elements and processes to achieve it played an important role in the design of the service. The component of FLAG that performs all these tasks includes a scheduler that will initiate a process that selects, from a list of all RW variables, those that must be updated "now" and, for each one of them, invokes the appropriate retrieval agent to obtain the data. Of course the system also determines the times at which every variable must be updated: when at least one customer needs its value to be refreshed.

The client, on the other hand, will determine, for every one of his real world variables, the minimum frequency with which he needs its current values, since it will probably have an effect on the cost of using the service. As for many other operations to be performed by a client, FLAG already offers some components which will help him determine these parameters, and others are either being designed or contemplated for future inclusion. Of course, some of the ingredients in these decisions are not entirely obvious or very difficult to generalize, since the actual benefit of knowing something in a particular business situation is variable, and often quite subjective.

Cost of processing the models

The other costs which must be minimized, within constraints, are all those associated with the processing, especially the executions of the models. Thus, the system includes many ways to reduce these costs, not only by avoiding unnecessary executions, but also through the use of efficient data structures to store and access the data during executions. Specifically, the models are designed to reduce the number of records to be processed. Furthermore, plain files are used instead of data base tables to store formulae, data values and other data involved in the execution processes.

The algorithms also must be designed taking into consideration that, though processing is extremely fast even in small servers, many models will be executed, should FLAG be successful, meaning it had many clients. The program that executes a model contains two ways to reduce the processing times: sub-models and additive operands of formulae.

Of course the main reduction is due to the use of sub-models, described above, since only a few of the model's calculated variables are computed. Additionally, the possibility to identify additive operands of formulae will allow the calculation of certain variables of a sub-model to be based on the change of one of its operands, rather than having to compute the formula as such. The Appendix contains an example and some comments about this feature.

Cost of alarms

FLAG was designed for Mexico, where many people lack the appropriate information channels and, on occasions, the training to use them properly. Additionally, the cost of certain telecommunication services is very high, compared to similar costs in more industrialized countries. Thus, the costs associated with the transmission of messages can be significant, and the reduction in these costs can only be achieved by sending fewer messages. The determination of the parameters involved in the specification of his alarms criteria has to be based on a cost-benefit analysis, and some help is provided by the possibility to simulate certain changes in external variables, which in turn aid the client to determine when he should request an alarm.

How FLAG Addresses Quality of Information Attributes

As was stated previously, FLAG was designed as an Informing Service. Thus, it is appropriate to ask how, and to what degree, every attribute regarding the quality of the information and the informing process were achieved by the design and implementation of FLAG's elements. Table 12 clearly reflects that both parties are asked to participate actively in the informing process to obtain higher quality levels. Therefore, the designers constantly have to change roles from one side to the other, a practice that is well known to system designers in any context.

Table 12: Client and Provider's roles in FLAG to achieve QI attributes

INFORMING CLIENT	ATTRIBUTE	INFORMATION PROVIDER
Specifies frequency of updates of his RW variables.	**Current**	Invokes update mechanisms with the required frequency.
Alarm criteria. May query his model anytime.	**Timely**	Sends alarms
Formulates a model.	**Interpretable**	Delivers information via the model.
Only asks for what he can use.	**Relevant**	Does not communicate *useless* data.
Asks for everything he needs.	**Complete**	Ensures that changes are detected every time they occur.
May check a vital item with own sources.	**Correct** **Credible** **Reliable**	Uses credible sources; tries to avoid mistakes; compares several sources.
May indicate own sources to the provider.	**Rarity** **Exclusivity**	May offer exclusive use of certain data to a subset of clients.
Determines the degree of confidentiality of his data.	**Secret**	Restricted access to the client's data and model. Scrambling of data and messages.
Limits number of executions of his model. Does not ask for superfluous information.	**Cost effective**	Offers data to many customers. Chooses sources by cost. Automates almost all tasks.

A careful study of this table gave rise to two quite different remarks. First, the reliability and credibility attributes are not only critical, but are probably not sufficiently addressed by FLAG. Gackowski (2005) formulates it in a way that leaves little to add: "Since credibility is rarely-to-

never fully attainable, in many situations, users must learn to act with only an acceptable level of credibility labeled **actionably credible.** For practical purposes, ***actionably credible*** can be defined as the degree of credibility at which the user is willing to take action." Perhaps the descriptions of the service provided in this paper did not furnish all the answers one could expect. The author's conclusion was that, of course, there is no way to provide total reliability, but the service should invest heavily in trying to avoid both disinformation and misinformation. All possible circumstances that might cause a customer to receive wrong information should be taken into account. This requires research and imagination. As a consequence, devices of all sorts, including double checking, comparison of several sources, human criterion applied to strange looking data and others – here is where we invoke imagination – must be included to the degree cost-benefit considerations will allow.

The inclusion of random variables in the model is one way of addressing this problem. If a farmer asks FLAG to provide him with the best prediction of the price of corn 7 months from now, it might be a good idea to gather information from several sources and construct a probability distribution, which should reflect the degree of accuracy of each of them. Of course this will not remove all factors that may make the prediction turn out to be wrong, but perhaps it furnishes some tools for a clever modeler to be aware of and handle the risk involved in such predictions. Thus, he will decide if he can act upon the new information (i.e. the information is actionable for him).

The other remark is of a totally different nature. There is something missing in Table 12: the complexity issue. If the potential customers cannot deal with the functions and concepts involved, the service might be very well formulated, but it will not be a service at all, since the latter designation implies that it actually *serves* somebody.

After citing Cohen (1999): "The driving force behind the creation of informing environments and delivery systems is that a task needs to be accomplished," Gill and Hicks (2006), state: "As the driving force in the development of informing systems, the role played by task is of particular significance to the individuals responsible for developing such systems, since better insights into the task to be accomplished should improve their understanding of the resources that will be required. In this context, task complexity appears to be a particularly important characteristic." Gill (1996) exposes a factor which adds another dimension to the design: "When IT is introduced, the situation often becomes even more convoluted – as such technologies can enable task performance by new individuals." Thus, when alternatives regarding the ways to offer FLAG's features to its clients were evaluated, we took into consideration several factors which contribute to the complexity of the associated tasks. Since they were used in this evaluation, we quote some of the 13 "existing task complexity constructs" presented in table format on page 8 of the above quoted paper by Gill and Hicks (2006), where the numbers correspond to the ones provided in the authors' table: 1) Degree of difficulty; 3) Degree of stimulation; 4) Amount of work required to complete the task or information load associated with the task; 5) Amount of knowledge; 6) Size; 7) Number of paths; 8) Degree of task structure; 9) Nonroutineness or novelty; 12) Function of alternatives and attributes.

To deal with this complexity and its probable implications, FLAG offers several versions of the programs described in this paper, especially for the more complex tasks. In fact, I do this in most of my information systems and software products: the user will either choose or be assigned an expertise level, and the programs will take this level into account, offering options or alternative paths only when they will not confuse or affect the user in some other way. The extra effort has very often paid off: the systems were easier to install, operate and of course, learn by its users. As an example of such versions of the programs, two different versions of the program to formulate alarm criteria are shown in the Appendix.

During the design of the features to be included in FLAG's components, the resulting complexity of the associated tasks always had a high priority. For example, the description of the alarms criteria might elicit a remark in the sense that they are unnecessarily complex and sophisticated. The main object of their design was to balance two sometimes conflicting attributes of such criteria: they should be powerful, or as we like to call it, flexible, but they must also take into account the resulting complexity of the tasks involved. Comments about the definition of alarm criteria can be found in the previously cited paper by Bauer Mengelberg and Velazquez (in press). A brief remark regarding the use of sub-totals to replace parenthesis was made when the formulae were introduced. However, in the design of all interfaces which could be used by clients, as opposed to those available to the employees of the service, who can be trained to deal with even complex tasks, the balance between complexity and flexibility was determined after evaluating several alternatives. Since the most frequent use of the system by a customer will probably be to query his data, special efforts were built into the programs that will result in the delivery of information. In the Appendix, a description of how a query can be defined and used is furnished via screenshots of the main interfaces. The same was done for the program to specify the formula of a calculated variable. Since the definition of the alarm criteria appears to require some additional explanation, some details are also presented in the appendices.

The principles and theories used to determine the most favorable – usable – designs were primarily based on my experience as an Information Systems developer, where these components often are the factors which will determine the success or failure of a system. Since I apply a simple criterion to determine the degree of success of my systems, namely their permanence, apparently some of these choices seemed to have been correct. Nowadays there is a lot of published research on these topics, but when I needed such help most, in the early 70's, only few papers were devoted to the subject and they could not replace a very thorough study of the future users of the systems, especially their individual characteristics and attitudes. In the present situation, the following questions were asked again and again: will the potential customers be able to formulate their models, establish the different criteria, use the structure of FLAG's formulae, benefit from the possibility to include random variables in their models, and lastly – but perhaps crucially – will they be able to use the queries - the "end product" as far as the clients are concerned?

Conclusions

This paper addressed two related, but quite different, topics. The first is a description of the objectives and design of the informing service called FLAG. But it also illustrates the additions and improvements to the informing process involved, as a result of viewing it in the IS context, particularly Gackowki's purpose-focused framework, imposing information quality criteria, such as it being actionally timely available, and the roles played by both parts of the process. The data models and the main features of the informing service were described, with comments intended to explain them, rather than to furnish details, which were only added when they somehow contributed to the determination of the degree in which several quality attributes of the resulting information delivered to the client, were achieved. FLAG is constantly being modified, not only through changes and completion of certain components, data models and the way the functions are performed, but also by the addition of new features to improve the quality of the service to its clients. The design of the pricing and marketing aspects is in its initial phase, as well as many of the programs which will allow customers to formulate their models and change their own variables' values on the Internet. Future research includes the use of more powerful and flexible mathematical models, and connecting the models to other systems in both directions: to obtain values determined by these other systems, and to provide the newly acquired information to others such as decision making processes. Finally, can FLAG be useful? After an informal survey of several farmers and owners of small agribusinesses in an area surrounding our campus in the State of Mexico, several things became apparent. They were shown prototypes of some of the

functions, complemented with a live session based on a simple example, which provided details and the means to answer questions and address doubts. The possibility of having such information was unanimously considered important, or even vital. The tasks it would involve and the yet undetermined costs associated were obvious drawbacks. Those who understood the concept of an alarm being issued should one be necessary – not all those interviewed - were delighted by this feature, but the enthusiasm dampened considerably when told how they would define the rules that would result in such warnings. This of course was predictable, despite the informal and elementary explanations of the tasks involved. Most of the persons interviewed expressed a certain surprise regarding the ease with which they could update their own data. When asked to try to find out the values of some of their variables, a process we refer to as queries, but which they called questions, the surprise turned to disbelief since they performed tasks which, a few minutes earlier, they considered well beyond their possibilities. Thus, FLAG probably has a chance as a marketable service. However, perhaps due to the fact that its administrative components are not ready, and the pricing system is not available, nobody has yet decided to consider actually offering this service.

References

Array formulae. (2008). Retrieved April 4, 2008 from http://www.decisionmodels.com/optspeedj.htm

Bauer Mengelberg, J. R. (2005). Teaching system access control. *Issues in Informing Science and Information Technology, 2*, 139-158.

Bauer Mengelberg, J. R., & Velázquez, G.D. (in press). Timely informing clients of the impact of changes in their business environment. *Issues in Informing Science and Information Technology, 7*.

Birdsall, W. F. (2009). The role of the client in informing science: To be Informed and to Inform. *Informing Science: the International Journal of an Emerging Transdiscipline*, *12,* 147-157. Retrieved from http://www.inform.nu/Articles/Vol12/ISJv12p147-157Birdsall535.pdf

Cohen, E. B. (1997). IS as an evolving field. *Proceedings of the World Multiconference on Systemics, Cybernetics, and Informatics, Caracas, Venezuela*. *July 7-11 1997*, ed. by N. Callaos, C. M. Khoong, and E. Cohen.

Cohen, E. B. (1999). Reconceptualizing information systems as a field of the transdiscipline informing science: From ugly duckling to swan. *Journal of Computing and Information Technology*, *7*(3)*,* 213-219.

Excel's Smart Recalculation. (2008). Retrieved April 6, 2009 from http://www.decisionmodels.com/calcsecrets.htm

Gackowski, Z. (2005) Informing systems in business environments: A purpose-focused view. *Informing Science: the International Journal of an Emerging Transdiscipline*, *8,* 101-122. Retrieved from http://www.inform.nu/Articles/Vol8/v8p101-122Gack.pdf

Gill, T. G. (1996). Expert systems usage: Task change and intrinsic motivation. *MIS Quarterly*, *20*(3), 301-329. Retrieved February 15, 2010 form http://www.misq.org/archivist/vol/no20/issue3/vol20n3art3.html

Gill, T. G., & Bhattacherjee, A. (2007). The informing sciences at a crossroads: The role of the client. *Informing Science: the International Journal of an Emerging Transdiscipline*, *10*. 17-39. Retrieved February 20, 2010 from http://inform.nu/Articles/Vol10/ISJv10p017-039Gill317.pdf

Gill, T. G., & Hicks, R. C. (2006). Task complexity and informing science: A synthesis. *Informing Science: the International Journal of an Emerging Transdiscipline*, *8*, 1-30. Retrieved February 20, 2010 from http://www.inform.nu/Articles/Vol9/v9p001-030Gill46.pdf

Gupta, V., & Prasad, S. K. (2005). *Hierarchical data structure tree building and state propagation using common information model*. Fort Collins, Co. USA. Retrieved March 6, 2008 from http://www.freshpatents.com/Dynamic-hierarchical-data-structure-tree-building-and-state-propagation-using-common-information-model-dt20051215ptan20050278354.php

Stahl, B. C. (2006). On the difference or equality of information, misinformation, and disinformation: A critical research perspective. *Informing Science: the International Journal of an Emerging Transdiscipline, 9*, 83-96. Retrieved March 3, 2010 from http://www.inform.nu/Articles/Vol9/v9p083-096Stahl65.pdf

Ulrich, W., (2001). A philosophical staircase for information systems definition, design and development: A discursive approach to reflective practice in ISD (Part 1). *The Journal of Information Technology Theory and Application (JITTA), 3*(3), 55-84.

Weiss, M. A. (1993), *Data structures and algorithm analysis in C*. Redwood City, California: Benjamin/Cummings.

Wiener, N. (1954). *The human use of human beings - Cybernetics and society*. Garden City, New York: Doubleday Anchor Books.

Appendix
Additional Description of FLAG

Introduction

To complement the descriptions presented in the paper, some of FLAG's functions are illustrated and explained. However, it is not a full description of FLAG; only a few screen shots are depicted, and explanations provided when they were considered necessary to achieve the desired level of detail. Please note that:

- Several "screens" were modified for the purpose of their inclusion here: the layout was changed to achieve a reduction in size (most of the actual screens occupy the entire window), but especially the font sizes were increased for legibility. Additionally, many command buttons, operating instructions, help functions and additional features were omitted
- Variables and their values used to populate the screens do not correspond to the cash flow example used in the paper.

A MENU Distributes the Main Functions

After the proper authentication by the user of the current session, he is offered the main menu, depicted in Figure 6. Most of the functions invoked are explained in this Appendix. The "calculate levels and sub-models" function is technical, and will be described below. Of course there is a complete help facility, with explanations and instructions to enable a client to use the system. In all other functions, instructions and exception handling explanations are included in the system, but are not shown on the screen-shots presented here.

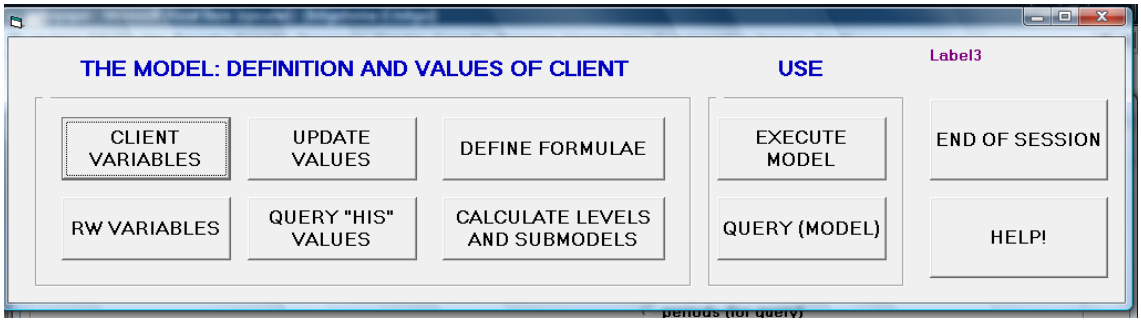


Figure 6: The main menu of FLAG's client programs.

The Client Defines or Updates his Model

The client introduces his "client variables" (his own), chooses some of the Real World variables offered by FLAG, indicating the periodicity with which he needs their respective updates, and defines his calculated variables, indicating their formulae. An outline of the way this is done is shown in the following sections.

Client variables of the model

Figure 7 shows a prototype of the form used to update a client variable. If the variable is a calculated variable, the "formula" frame is shown so it can be defined at that moment, but usually this will be done for all variables with a special function. The same form is used to update or query the client variables of a model. Texts shown in blue represent display fields: they result from the formulae of the model. "Formula to dad" is a FLAG feature: if the variable is an additive operand of a variable, the latter becomes its dad (parent) and a + or – sign indicates this fact. As will be seen later, this is useful when sub-models are executed.

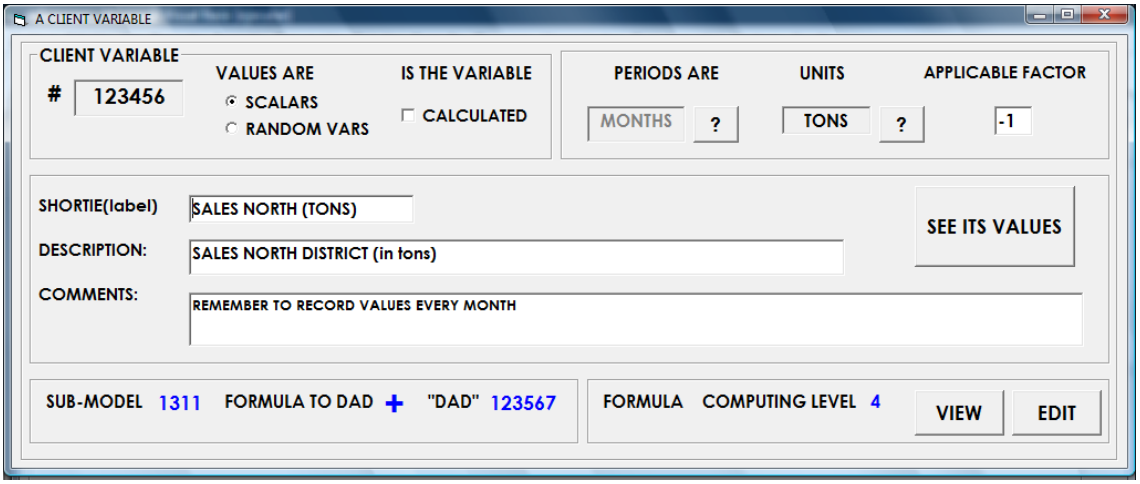


Figure 7: A client variable and its main data fields.

Real world variables included in the model of the client

From a list of RW variables offered by FLAG, the client selects those that will furnish the necessary information about his business environment. For each of them, the desired update frequency is indicated (the forms to do this are not shown). Of course, a client may request other variables – not offered at that moment – and FLAG will determine if it can offer them or not.

Synonym variables, those that are used as operands but share the values of the variables they represent, are created automatically by the program that updates formula, whenever a variable is used as an operand of more than one formula. However, the system offers a way to introduce them manually if the user desires to do so, mainly to plan his formulae carefully before introducing them in the model.

A technical observation: since their values are stored in separate files, variables are numbered in ranges in FLAG. Client, synonyms, real world values have their own ranges, which are subdivided into those for scalar and random variables, that is, each of them have a range of their own. For example, the client variables use numbers from 1 to 1,000.000, but scalars cannot have a number greater than 500.000.

Values of client variables

The values of the client variables can be updated anytime using a form such as shown in Figure 8. From a list of his variables, the client chooses a variable and can change the value of any of its periods. Though this will not usually be the case, he may also update the values of the RW variables he uses, though only in a copy of the "true" values, those updated by the service itself. This function was included for simulations or "what if" uses of his model.

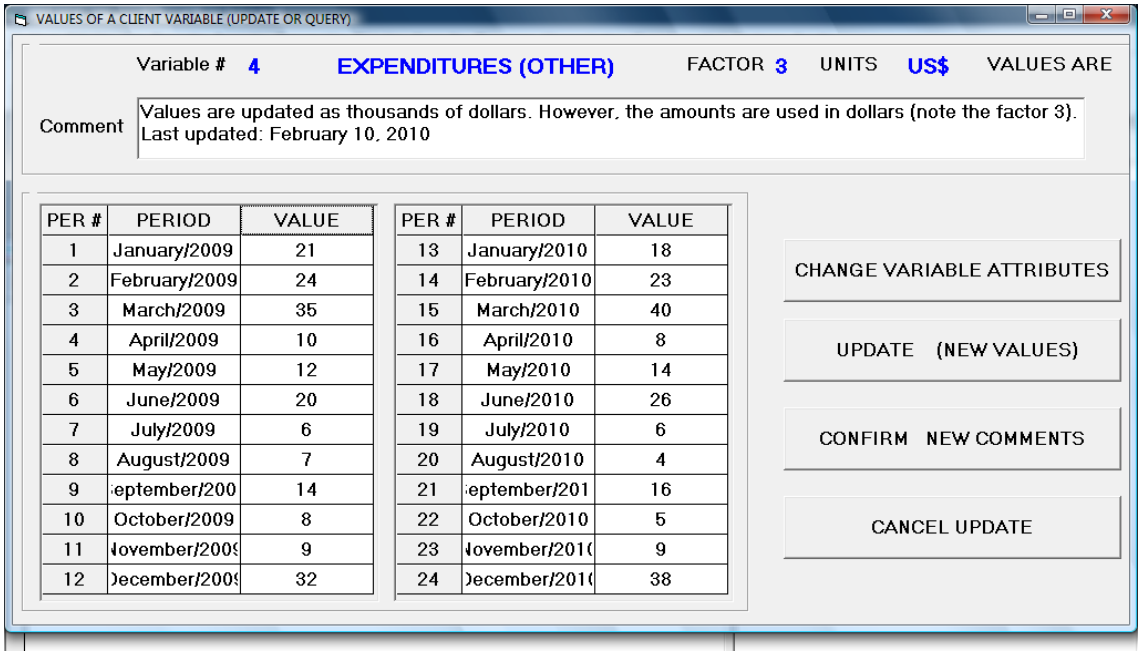


Figure 8: Form to update or query values of a variable.

Formulae and their validation

The forms used to define formulae were described in the paper. A formula can be tested to determine whether it reflects the calculations necessary to achieve the variable's purpose: This feature is illustrated with the following example.

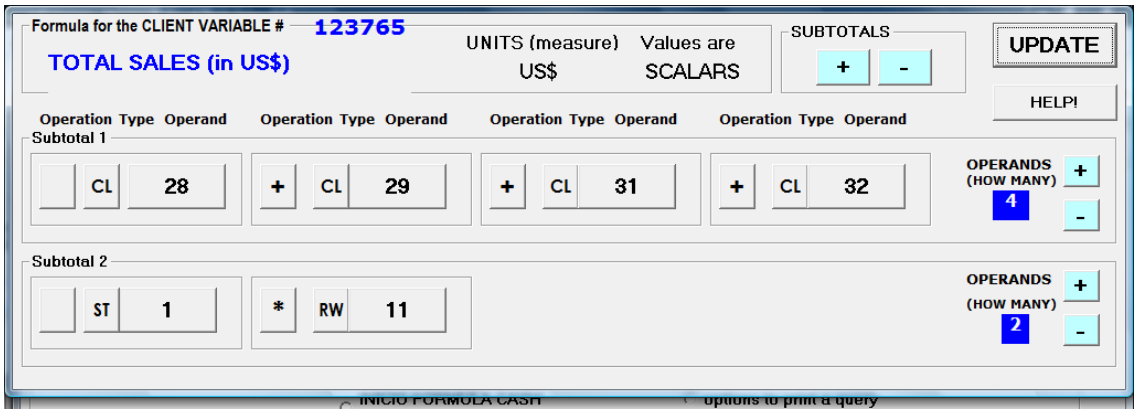


Figure 9: The formula for the variable to illustrate the testing procedure.

Suppose the model includes client variables for the volume of sales (in tenths of a ton) in the 4 sales districts N, E, S and W, numbered respectively #28, #29, #31 and 32, and the RW variable

R11 – PRICE PER TON in US$ /Ton is included. The formula for TOTAL SALES shown in Figure 9 is V(123765) = (V28 + V29 + V31 + V32) * V11.

As depicted in Figure 10, one may indicate which formulae of a model are to be tested: that of an individual variable, those of a sub-model, or even every formula of the model. For the RW variables, the process will use the values he has at that moment on his computer, a copy of the true values. Although they might not be the last values, they will serve the purpose of validating his formulae.

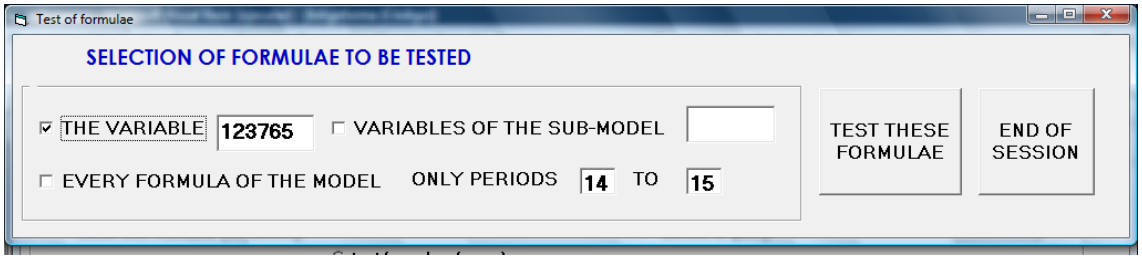


Figure 10: Testing one or several formulae.

One by one, the formula for the chosen variables are displayed. The values of their operands (for the selected periods, since checking the formula for some of them may be sufficient) and the resulting values are shown. There is a direct link to the program to update a formula, as can be seen on Figure 11, which depicts the form used to show the formula and its effect on the calculated variable.

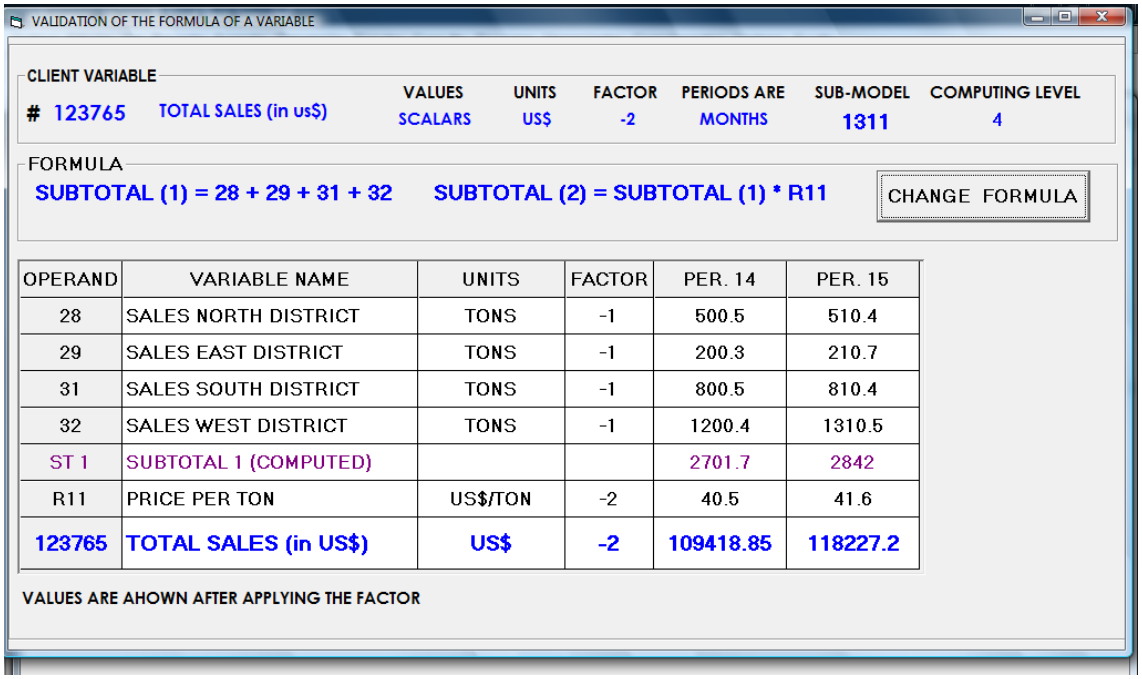


Figure 11: Validation of the formula of a variable.

Sub-models and levels of the formulae

Whenever variables or formulae of a model are changed, a process that updates the levels of the formulae must be executed. As pointed out in the paper, the order in which formulae are computed during execution has to be such that all operands of a formula are computed before their use in the formula. This is done assigning computing levels to the formulae: the level of a for-

mula is “one more” than the level of its operand of highest level. A similar process updates the sub-models: they are subtrees of a node of the tree representing the model. For every node, the sub-model of its “sons” (operands) are built concatenating the sub-model of the node and a consecutive number, in the order they appear in its formula.

The sub-models involved in the cash flow example were shown to be those reflected in Table 13 (the indication of “irrelevant” means it has no bearing on the explanation, since the formulae are not involved in the calculation of the sub-model).

Table 13: Sub-models of certain variables of the cash flow example

Variable Id	Name	Sub-model	Formula	Formula-to-dad
#1	Cash flow	**1**	#2 - #3 - #4	
#2	Profit	**12**	#5 - #7	+
#5	Revenue	**121**	#6 * RW#1	+
#6	Harvest	1212	Irrelevant	
RW#1	Market price	**1211**	---	
#7	Costs	122	Irrelevant	-

Suppose there is a change in the value of RW#1 (MARKET PRICE), and the corresponding sub-model is “1211”. Only the variables whose sub-models are “121”, “12”, and “1” must be recomputed. Additionally, since Variable #5 is an additive operand of the formula for variable #2 – note the value of “formula to dad” – instead of computing the formula for PROFIT, the change in value of every period of Variable #5 is added to the value of the corresponding period of variable #2. The same is true when determining the new value of variable #1, since #2 is an additive operand of its formula.

If only the models of a few clients must be executed, these reductions will be hardly perceptible, especially if the models are small (to quantify this, let’s say that they have less than 100 variables). But if FLAG had a few thousand customers, and at a given moment had to execute 2000 models simultaneously – or one after another, a difference of 80% in execution time will be of great value to the service. One might safely say that for large models, the reduction in execution time is much greater. Though we have not had a chance to try really large models- with thousands of variables - on the present generation of computers, previous experience with these models (which I first used 35 years ago in a very large company on IBM 370 computers) shows that execution times simply disappear when only sub-models are computed.

Queries

A client may query his model anytime, as long as he is connected - usually via Internet - to his database and data value files. However, there are other possibilities: he may request that updated values of his RW variables be sent to him, so he can use his own computer (and programs installed on it) to query or use his models otherwise, typically for simulations.

Many options are offered: he can specify queries, store them for future use (their definition, not the results) or define a specific query for a unique use. Typical queries are: a sub-model, a range of periods for all variables (or those he indicates in several ways, such as filtering with one of many criteria offered or even by enumeration of the desired variables). A few features of the pro-

grams through which the client can exploit his information are described in the following sections.

Selection of the query to be executed

As shown in Figure 12, a list of all stored – catalogued – queries is presented when he invokes the corresponding module.

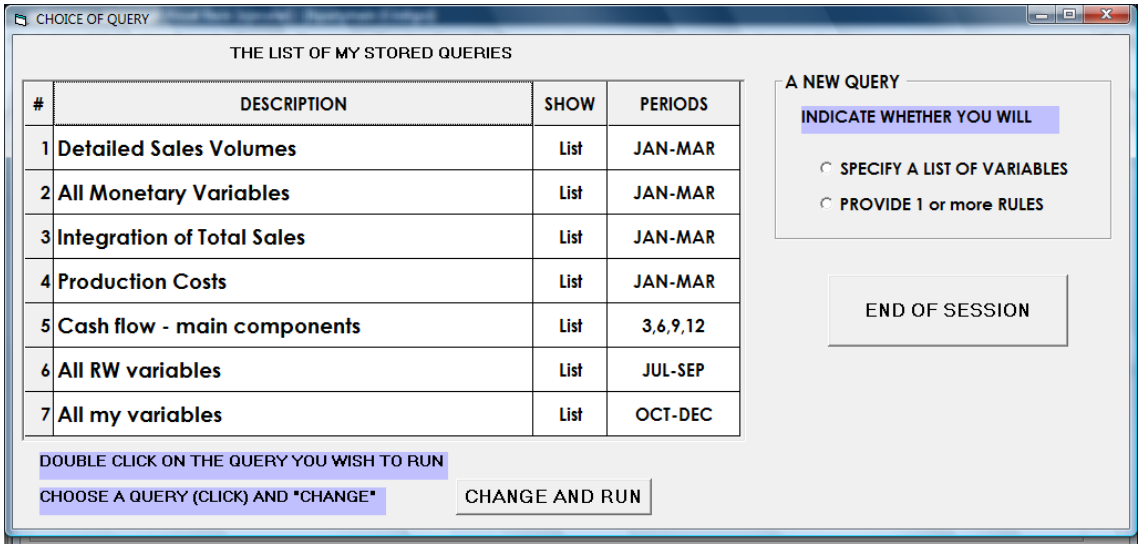


Figure 12: Selection of the query to be modified or executed.

He chooses one of these queries, or he may indicate that he wishes to formulate a new query. In the latter case, he is asked how he will define it (because the selection will result in different ways of prompting the introduction of the variables to be included as part of the query).

Definition of a query by way of listing the variables desired

To produce a list of variables, the user can add variables one by one to a list, and reorder or delete them. Lists of variables of his model are offered for this purpose, should he need them, since he can include them indicating their variable number. This form is not shown.

Definition of a query using selection-of-variables rules

We call a criterion for the inclusion of several variables in a query a *rule*, such as "variables that are measured in Tons." If rules are to be used, one or more of these will be included using the form depicted in Figure 13, where the list was populated to provide examples of rules one might specify (though of course a particular query would not have such a great variety of rules).

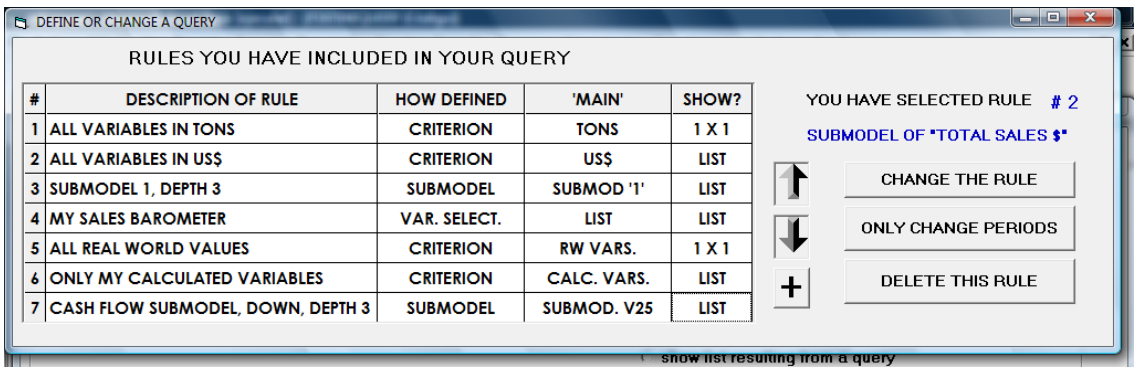


Figure 13: A list containing examples of the rules specified for a query.

Each of these rules is defined on a form on which the corresponding values are indicated. Figure 14 illustrates an example of a rule: include “all variables”(RW and client variables) starting with the variable with sub-model 1311, “upward” at a depth of 3. This would show the variables with sub-models 1311, 131, 13 and 1.

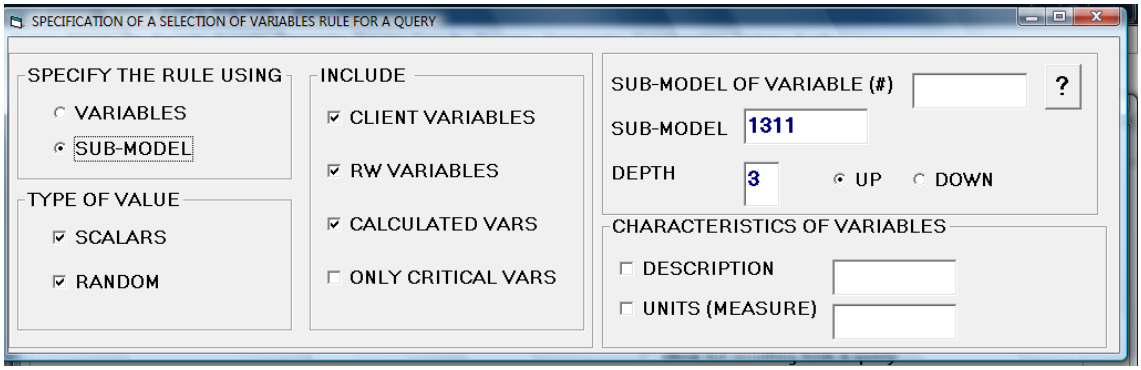


Figure 14: An example of the specifications for a selection-of-variables rule.

Though in Figure 14 both alternate frames are shown, when running the program only the one corresponding to the selection of how the rule will be specified will cause the corresponding fields to appear. For example, in the example, since “sub-model” was chosen, the options to specify “characteristics of variables” would not appear on the screen.

Note the checked options (all variables, including those with scalar or random values). If an option is not checked, the variables corresponding to that type will not be included in the query. For example, selecting “only critical vars” will not show calculated variables that were not designated as critical.

The “**?**” beside a field in FLAG indicates that a list of values of that type will be displayed. In this form, it would show a list of all variables that satisfy previous conditions specified (if only RW variables were checked, the list would only show the real world variables included in the model).

The periods to be shown

For all queries, the periods to be shown, as well as the mode in which the data will be displayed, must also be indicated. The form, depicted in Figure 15, adjusts to the selected option: the option “range” prompts for initial and final period to be shown, whereas “list” will allow the introduction, one after the other, of as many periods as were specified in the corresponding field. For example, only the periods representing the end of quarters are required for a particular use – or an executive report.

As mentioned before, values are stored as integer numbers. However, a factor may be defined for a variable, such that it is applied to the value to determine the true value. For example, a value of **1234** of a period of a variable with a factor of **-2** would be shown as **12.34,** but if the factor were **2**, the same value of the period would result in 123,400. The option to display the values without applying the factor is mainly offered to check the values actually provided; this is especially useful when the client wishes to check the values of his own – the client – variables.

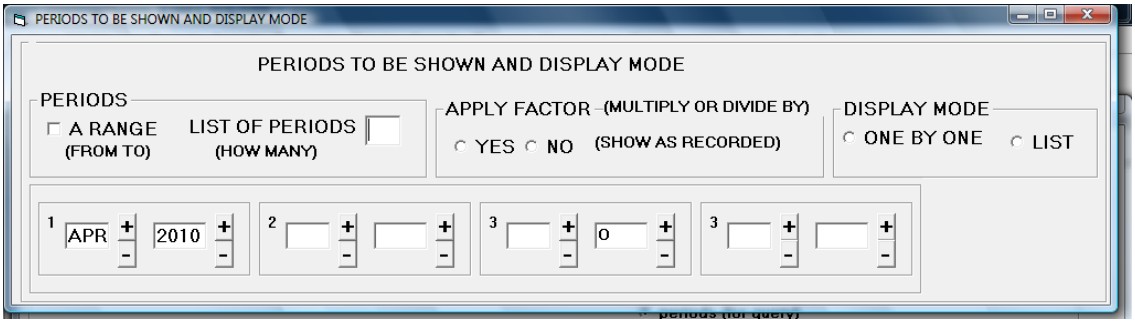
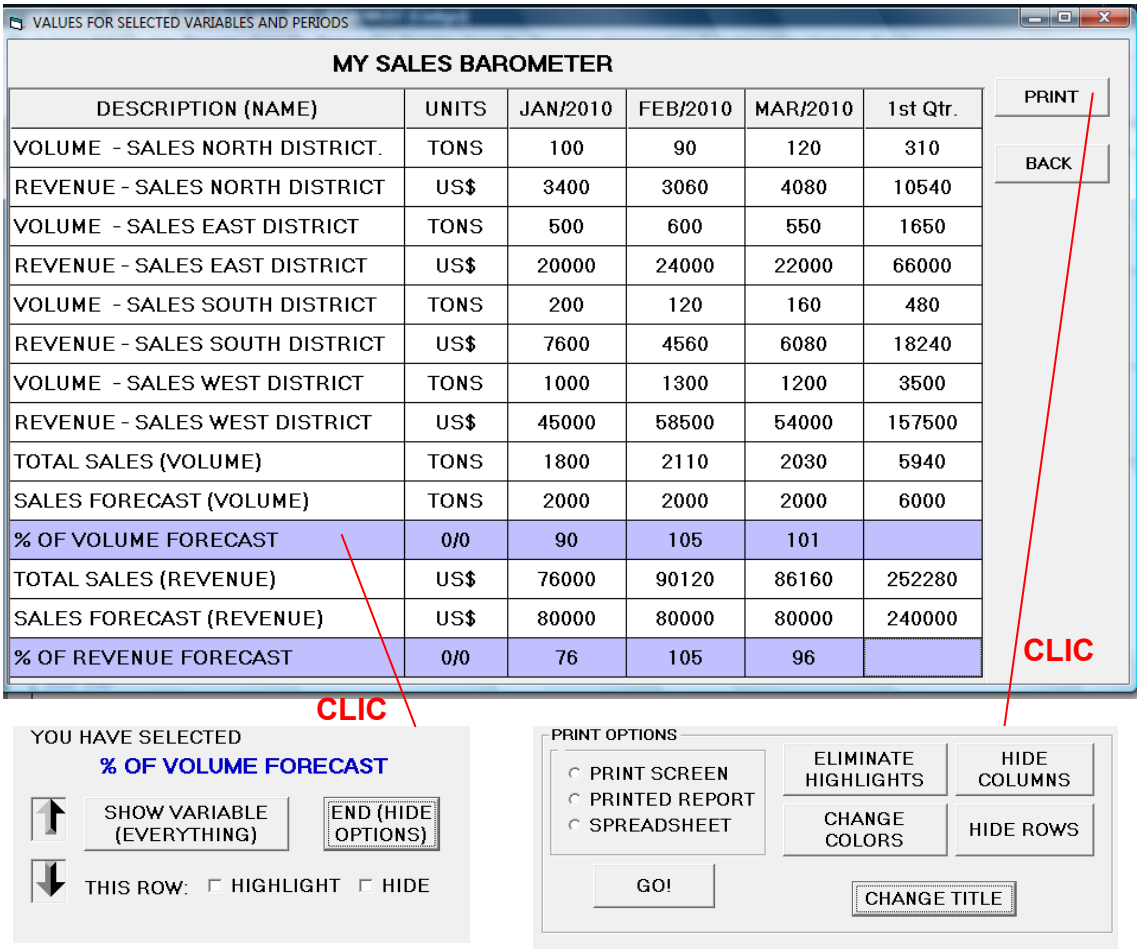


Figure 15: Form to define details of the query.

The result of executing a query

When a query is executed, the result may be a list of variables and their corresponding values for the selected periods, or the values of the variables included in the query will be displayed one after another. Figure 16 shows a prototype of a query resulting in a list of the specified variables. Periods are shown in columns.



DESCRIPTION (NAME)	UNITS	JAN/2010	FEB/2010	MAR/2010	1st Qtr.
VOLUME - SALES NORTH DISTRICT.	TONS	100	90	120	310
REVENUE - SALES NORTH DISTRICT	US$	3400	3060	4080	10540
VOLUME - SALES EAST DISTRICT	TONS	500	600	550	1650
REVENUE - SALES EAST DISTRICT	US$	20000	24000	22000	66000
VOLUME - SALES SOUTH DISTRICT	TONS	200	120	160	480
REVENUE - SALES SOUTH DISTRICT	US$	7600	4560	6080	18240
VOLUME - SALES WEST DISTRICT	TONS	1000	1300	1200	3500
REVENUE - SALES WEST DISTRICT	US$	45000	58500	54000	157500
TOTAL SALES (VOLUME)	TONS	1800	2110	2030	5940
SALES FORECAST (VOLUME)	TONS	2000	2000	2000	6000
% OF VOLUME FORECAST	0/0	90	105	101	
TOTAL SALES (REVENUE)	US$	76000	90120	86160	252280
SALES FORECAST (REVENUE)	US$	80000	80000	80000	240000
% OF REVENUE FORECAST	0/0	76	105	96	

Figure 16: An example of the results of a query.

Note that a "last" column was added. This is the only such feature offered in the current version: one may ask for the sum of all columns to be added and shown as an extra column, and provide a

"title" for it. Additionally, a row can be excluded from this total (when the corresponding sum would be nonsensical, as the sum of the percentages would be in the depicted example).

When invoked by the corresponding command, one of the two frames containing options or functions will be displayed superimposed on the form which shows the results. Should the user wish to see everything, including the values for all 24 periods, of one of the variables, he would be shown – on a different screen – the definition of the variable and its formula, should it be a calculated variable. The result of a query may be printed or sent to a spreadsheet. The hide option here will eliminate that row from the screen (this operation cannot be undone, one would have to run the same query again).

An example of a 1x1 query was not deemed necessary: it simply displays the selected variables for each of them included in the definition of the query.

Several other options and features are available to formulate and display queries. For example, one may want to see periods as rows, and certain variables as columns. Additionally, FLAG will have a reporting component, but it has not yet been developed. The possibility to export the data or a query to a spreadsheet partially compensates for this missing feature, though of course – contrary to the spirit of the service – the potential users would have to know how to use and exploit such devices.

Brief comments about some of the other components

Alarms

This component is described in detail in Bauer Mengelberg & Velázquez (2010). However, after showing, in Figure 17, the role of this component in FLAG and how alarms are generated after an execution of a model, the two main forms used to define an alarm criterion are shown, since they might shed some light on some of the underlying concepts, already described in this paper.

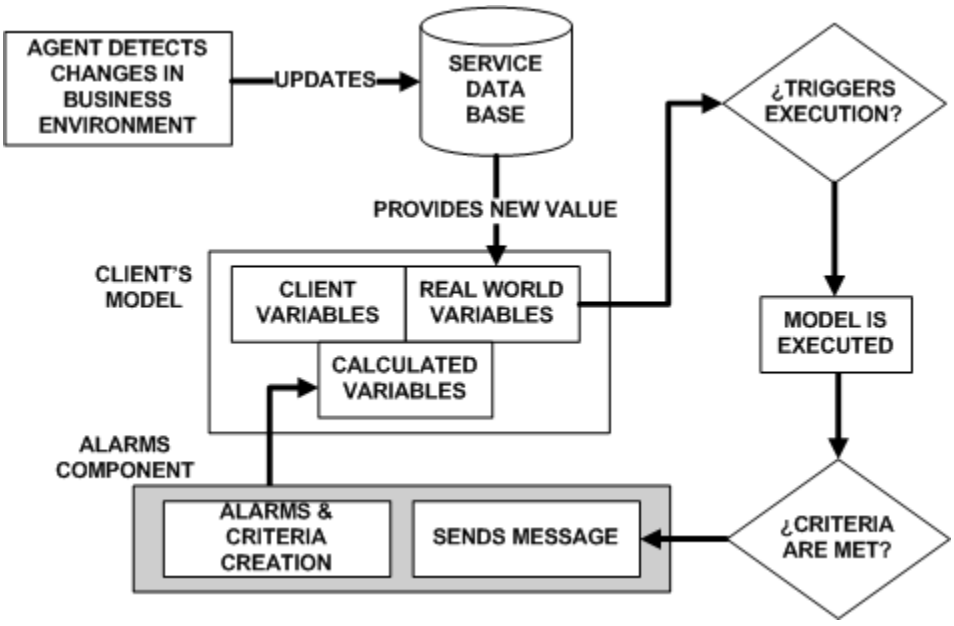


Figure 17: The ALARMS component of FLAG.

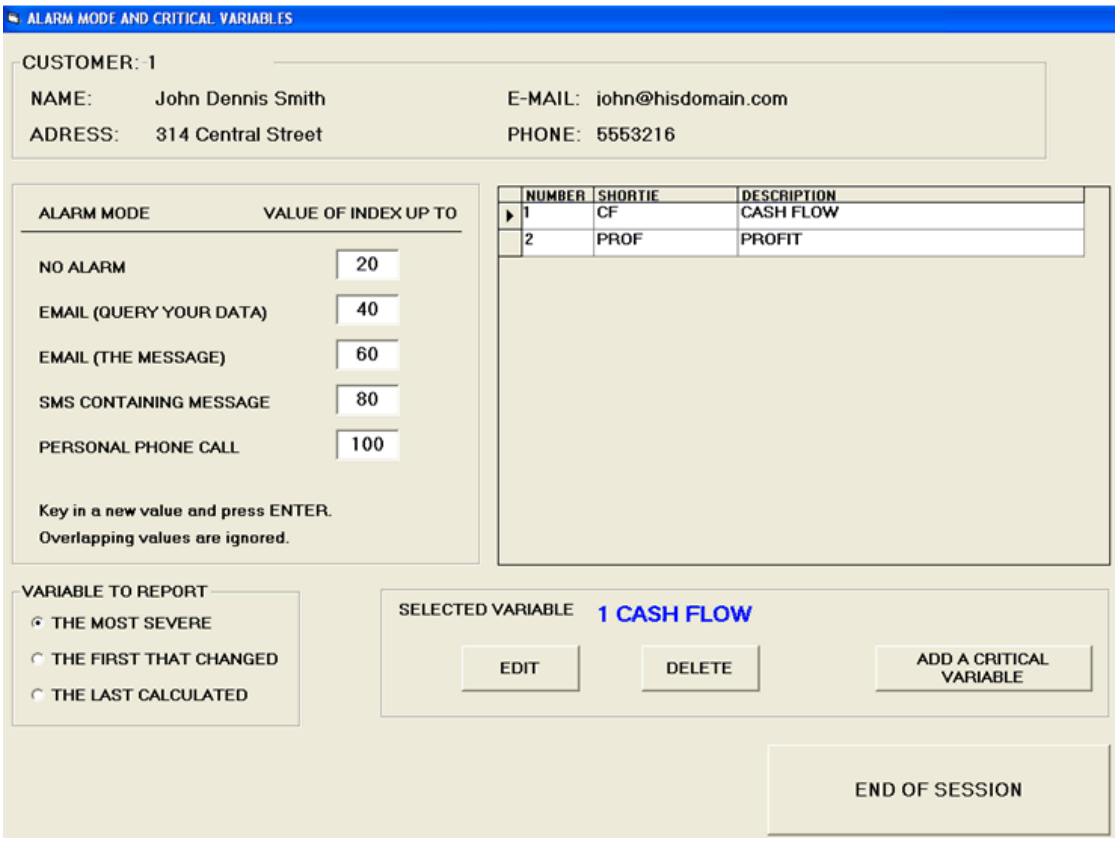


Figure 18: Selection of critical variables and communication modes.

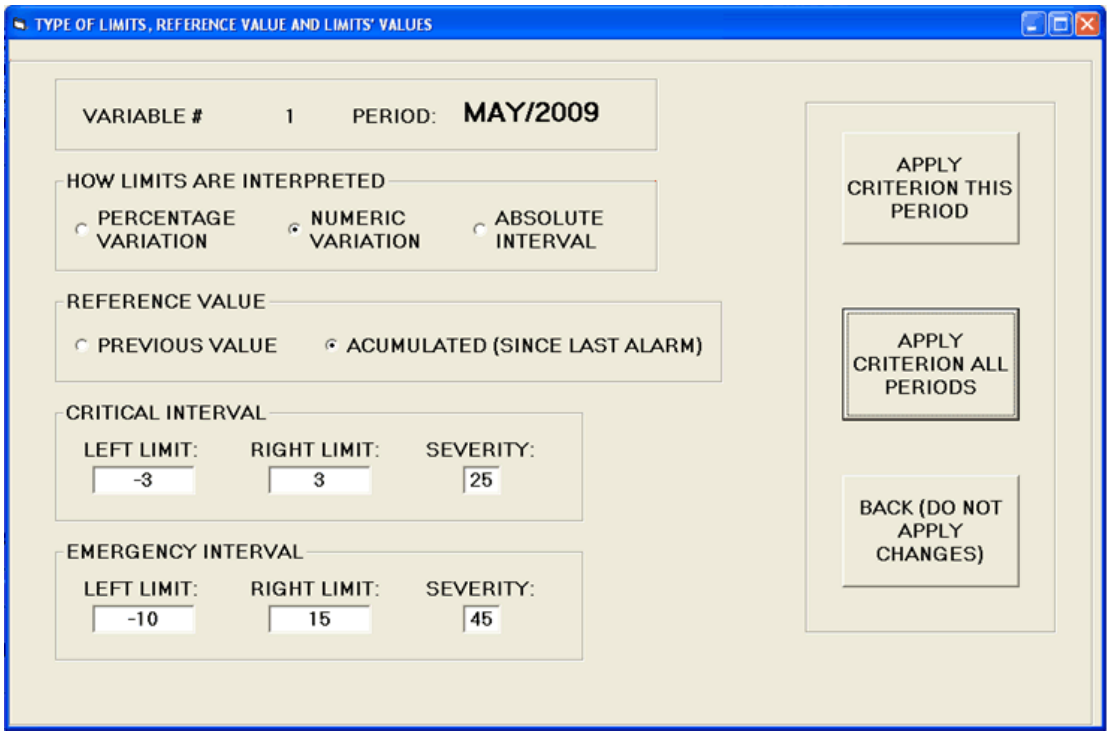


Figure 19: Form used to update the criterion's intervals for a period of the variable.

Using the form depicted in Figure 18, the client indicates which variables he considers critical (he will select them from a list of calculated variables). Besides, he will indicate the ranges that will, in turn, determine the way the alarms are delivered, according to the severity index.

Figure 19 shows the interface used to define the critical intervals, which are built around the previous value PV, unless the cumulative value (also called accumulated value) since the last alarm option is chosen. The limits of the interval will be PV – left limit, PV + right limit. These might be the numbers indicated, if the "Numeric Variation" option is active. If "percentage variation" had been chosen, these limits would be PV * (1 – left limit/ 100) and PV * (1 + right limit/ 100). For "absolute interval" the limits of the interval are precisely those indicated as such.

As announced, a typical "easy version" of the program to specify his alarm criteria by a client is shown in Figure 20. The form will not appear as such: the system reacts to his choices, and the consequent prompts will appear on his monitor. The choice of the variable (he can only use 1 such variable) will cause the periods to appear, and so on.

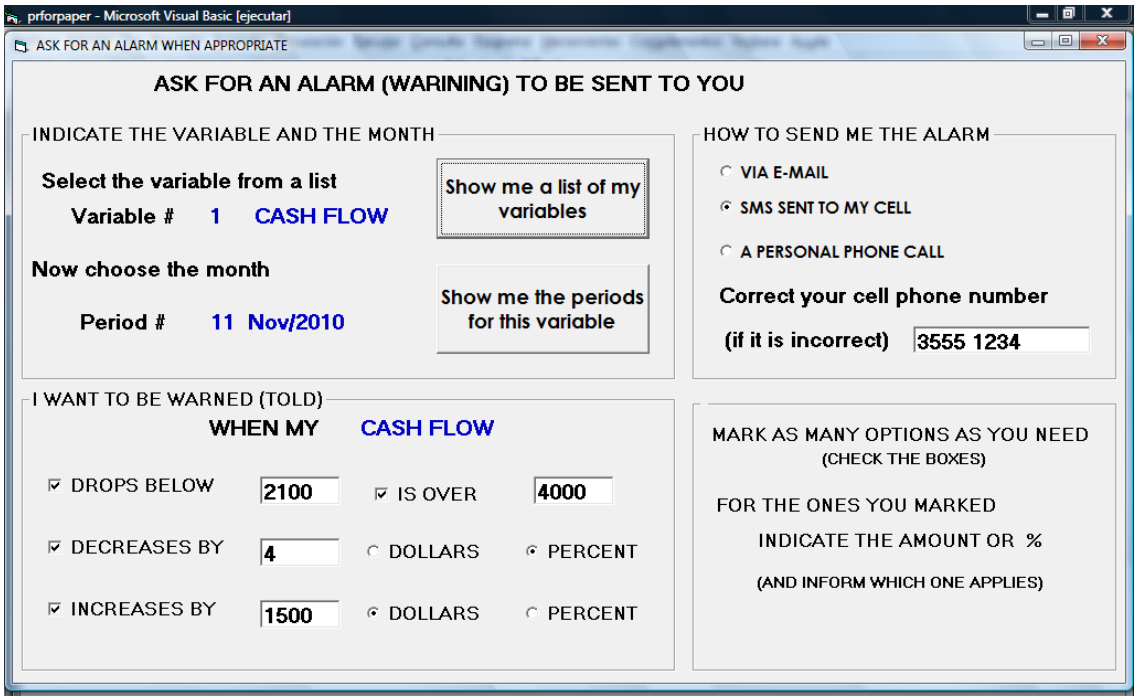


Figure 20: An example of a version for a non-experienced user to introduce alarm criteria.

The next version – for clients who have already mastered this one – would be to offer the same form twice (so they can select 2 critical variables). Of course the criteria he can define using this version are limited in several ways, but we let him think this was due to the designers: they just did not think of other options! Incidentally, based on the reactions from persons that have studied the alarms criteria, the opinion of many a reader of this paper might be that no such additional options are actually necessary, or even useful. The author, who designed FLAG, thought they were (useful) and thus, they will be offered to experienced clients.

How FLAG obtains current values for the variables offered to its clients

For every RW offered to its clients, FLAG determines a source of information, usually an Internet page. Rules to extract information using an intelligent agent are specified, and the periodicity with which it will be updated is determined according to the frequency desired by each of that

RW's customers; FLAG will usually use the most frequent one, unless availability or cost considerations indicate otherwise. A process, executed periodically (or even constantly) as part of FLAG's operation, will know which variables' values must be obtained at any given moment, and, using the rules to do so included in the specifications of each variable, will provide the necessary parameters for a retrieval agent to obtain the current value, which is recorded in a file. Other means to obtain current values include information brokers or even a manual update, where an employee will obtain a value and update the variable with a program provided precisely for that function. There is a component of the system which will combine the recent changes with all previous changes that might have a bearing on whether models of individual clients must be executed.

Executions triggered by changes of values of the RW variables

Since a client might not need his model to be executed whenever any change in his RW variables occurs, he is offered a way to avoid unnecessary executions, so as to save him money (the executions will quite probably carry an extra cost). The customer will indicate, in a way similar to the determination of when to receive an alarm, under what circumstances his model should be executed. Thus, whenever changes in RW variables are detected, a process will check these rules to determine if a particular model must be computed or not. Of course, the effects are cumulative: this is achieved by the use of an "amount-of-change" index, equivalent but different to the severity index of an alarm criteria.

When a model must be executed, FLAG has another component which determines if this process can be limited to the variables of one (or even several) of its sub-models. This is especially important if the number of clients is large, since execution times can be considerably reduced by not computing all the formulae of the model, as was explained previously. Unfortunately, no references can be provided regarding these topics, since the corresponding papers are still in the reviewing process of other journals; to make things worse, they were written in Spanish! However, some information will be posted on the previously cited site (http://jbauerm.com/flag/english/index.html).

Biography

Juan R. Bauer Mengelberg, after obtaining a degree in Mathematics from the Universidad de Buenos Aires, Argentina, got a PhD in Statistics and Operations Research from the University of Wisconsin, at Madison, where he also taught courses in the area of Stochastic Programming. He has since worked in Mexico, where besides teaching at the Colegio de Postgraduados, a school primarily involved in the field of Agronomy but which has both Statistics and Applied Computing departments, has held several positions, always connected with the field of Information Systems, in which he has also been a consultant all his professional life. He is primarily concerned with the subject of "systems that work", a concept he has extended to signify that they work even under abnormal circumstances. He has created and implemented many computer packages, and is currently working on several software products regarding publishing papers or books in the electronic media, and what he calls subsetting in very large data collections, specifically by means of a different type of database.