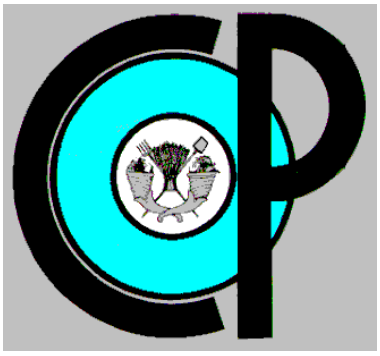


COLEGIO DE POSTGRADUADOS

INSTITUCION DE ENSEÑANZA E INVESTIGACION EN CIENCIAS AGRÍCOLAS

CAMPUS MONTECILLO
POSTGRADO DE SOCIOECONOMÍA, ESTADÍSTICA E INFORMATICA
COMPUTO APLICADO

Desarrollo Completo del Paquete de software "FLAG" (Flujo de Efectivo en Agronegocios)

RAFAEL VEGA RUIZ

TESIS

PRESENTADA COMO REQUISITO PARCIAL
PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS

MONTECILLO, TEXCOCO, EDO. DE MEXICO

2014

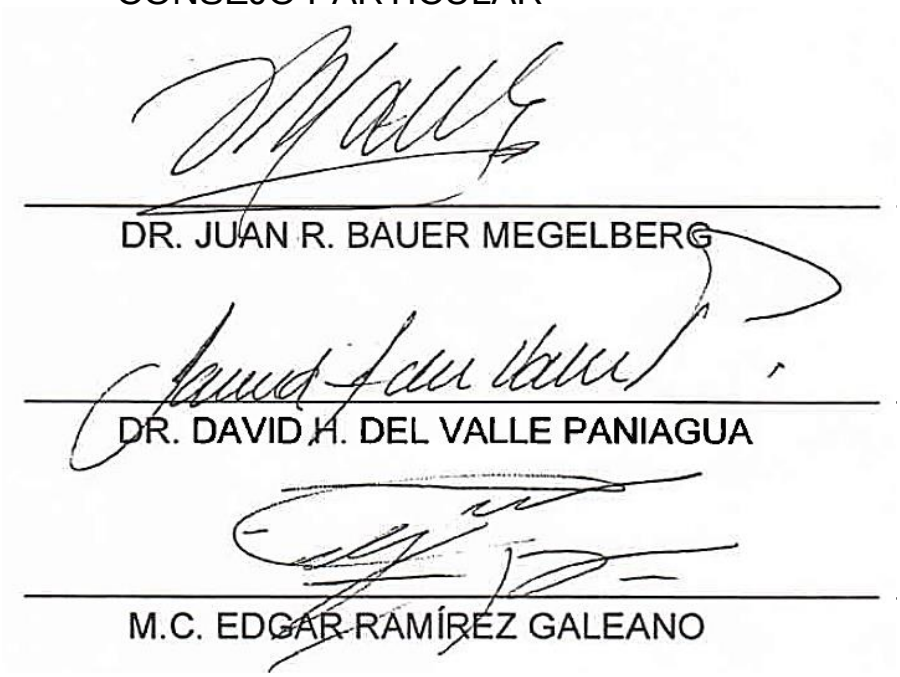
La presente tesis titulada **Desarrollo Completo del Paquete de software "FLAG" (Flujo de Efectivo en Agronegocios)**, realizada por el alumno Rafael Vega Ruiz, bajo la dirección del Consejo Particular indicado, ha sido aprobada por el mismo y aceptada como requisito parcial para obtener el grado de:

MAESTRO EN CIENCIAS

SOCIOECONOMÍA ESTADÍSTICA E INFORMÁTICA

CÓMPUTO APLICADO

CONSEJO PARTICULAR



CONSEJERO

DR. JUAN R. BAUER MEGELBERG

ASESOR

DR. DAVID H. DEL VALLE PANIAGUA

ASESOR

M.C. EDGAR RAMÍREZ GALEANO

Montecillo, Texcoco, Estado de México, 17 de Julio de 2014

DEDICATORIAS

A mis padres

Que siempre han estado conmigo en las buenas y en las malas, me dieron la vida y el sustento, me guían siempre por el buen camino con sus consejos y motivaciones.

Blanca E. y J. Luz

A mis hermanos

Porque siempre he contado con su apoyo y con toda su ayuda. Han sido una parte importante en mi crecimiento profesional.

Agustín, Orlando, Sandra y J. Alonso

A mis abuelos

Ya que han estado conmigo una gran parte de mi vida, me apoyaron y motivaron siempre en esta larga carrera profesional

Margarita y Joel, Natividad y Rafael †

A mis tíos y tías

No hay momento en que deje de pensar en sus palabras, ya que siempre me han dado aliento para continuar, para pensar bien las cosas antes de hacerlas, actuar de forma consciente y con prudencia para hacer de mí un hombre mejor en esta vida.

Sinceramente a todos

A mis amigos

A todos los que me acompañaron en esta aventura inolvidable e importante de la vida, a cada uno que me apoyó para la conclusión de este trabajo, a quienes me acompañan en estos momentos y a todos los que me brindaron su amistad para siempre.

A todos

AGRADECIMIENTOS

A la institución CONACYT

Porque con su apoyo tuve la oportunidad de realizar este gran logro en la vida.

Al Colegio de PostGraduados

Por darme la oportunidad de realizarme como profesional, como maestro en ciencias, forjarme en un hombre de pensamiento libre y crítico, por permitirme concluir este posgrado y hacer de mí la persona que soy ahora.

Al Dr. Juan R. Bauer Mengelberg

Por guiarme en la realización de este trabajo, su apoyo en el transcurso del mismo, sus invaluables consejos, por instruirme de una forma más que académica y así adquirir conocimientos valiosos en la vida.

A mis asesores: Dr. David H. del Valle y M.C. Edgar Ramírez

Por su constancia y apoyo en la realización del presente trabajo, por sus consejos y por los conocimientos que obtuve a través de ellos.

A mi familia

Por el apoyo moral, económico y académico que me han dado a lo largo de toda mi vida.

A mis amigos

Por el tiempo vivido en esta etapa de la vida, por su apoyo en momentos difíciles y compañía en los felices y más que nada por su invaluable amistad.

RESUMEN

El proyecto de investigación surge de un análisis del servicio de información denominado FLAG (flujo de efectivo en agronegocios) que consiste en que los clientes formulan modelos matemáticos basados en variables propias (de sus actividades) y de otras correspondientes a datos de su entorno de negocios. El servicio obtiene – con el uso de agentes informáticos – valores actualizados de estas últimas variables con la periodicidad indicada por cada cliente y recalcula las demás del modelo con estos nuevos valores. De ese modo el cliente no sólo obtiene los nuevos datos sino que puede determinar el impacto que tienen los cambios sobre su flujo de efectivo u otros indicadores que incluyó en su modelo. Este estudio resultó en la conveniencia de elaborar una nueva versión en la que se complementarían ciertas funcionalidades, pero en especial, se acortaran ciertos procesos para poder ofrecer el servicio a más clientes.

La versión nueva se implementó en la plataforma MS Visual Studio 10, en especial con el lenguaje de programación VB.net. Al utilizar la programación orientada a objetos se logró una reducción en los recursos informáticos, especialmente memoria, necesarios para ofrecer el servicio. En particular se modificaría el modo de ejecutar los modelos para minimizar los procesos de entrada y salida de datos y, en ciertos casos, evitar la ejecución de todas las fórmulas que componen el modelo cuando se pudiera determinar un subconjunto de las variables que se vieras afectadas por algún cambio en una variable del entorno (denominada variable del mundo real). Los agentes en la versión anterior se lanzaban consecutivamente, lo que producía cuellos de botella cuando la duración de uno de ellos era excesiva. En la nueva versión se utiliza programación multi-hilos para poder lanzar varios agentes informáticos simultáneamente.

El FLAG usa sinónimos de variables para que los modelos se implementen como árboles. En la versión actual esto es transparente al usuario, mientras que en la anterior el cliente debía especificar estos sinónimos, lo que causaba confusión y dificultad adicional. Las interfaces se mejoraron puesto que anteriormente implicaban conocimientos profundos del sistema y en ocasiones eran complejas en algunos aspectos. Para probar esta nueva versión, se efectuaron simulaciones que indican reducciones muy significativas en las duraciones de las ejecuciones de modelos, entre otras ventajas obtenidas. Este trabajo hace posible que haya interesados en ofrecer el servicio a alguna comunidad de empresarios de cualquier sector, a pesar de que se concibió para agronegocios.

Palabras clave: Servicio de información: modelos matemáticos; ejecución de submodelos; agentes informáticos: sistema de información.

ABSTRACT

The research Project arose from an analysis of an informing service called FLAG (cash flow in agribusinesses) based on mathematical models formulated by its clients, The models contain their own variables (reflecting their own data) as well as others that depend on the business environment. The service, through intelligent agents, obtains updated values of the latter with the periodicity indicated by every client, and computes the values of the other variables with these new values. Thus the client not only obtains the new values but also can determine the impact of these changes on his cash flow o other indicators he included in his model.

This review led to the convenience of preparing a new version in which certain features were completed but especially, the durations of processes were reduced in order to be able to offer the service to a greater number of clients.

The new version was developed with MS Visual Studio 10 platform, especially through the use of VB.NET programming language. The use of object oriented programming reduced the memory requirements to offer the service.

Particularly the process that executes the models was to be changed in order to minimize input-output activities, and in certain cases avoid recomputing all the model's formulae whenever it were possible to determine a subset of these that would be affected by a change in values of a single variable. In the previous version, agents were launched consecutively. This produced bottlenecks when one of them was slow responding. In the new version several agents are launched simultaneously by means of a multi-thread program.

Flag uses synonyms of variables so that the models can be represented as trees. In the new version this is transparent to the users, whereas in the previous one the client had to specify these synonyms, a circumstance that causes confusion and added difficulty to the formulation of the models. Many interfaces were improved as far as they now do not require a deep understanding of the topics and are simple and efficient to use. To test this new version, simulations indicate very significant reductions in the duration of the execution of models, among other advantages obtained were made. This work suggests that there will be interested in providing service to a community of entrepreneurs in any sector, although conceived for agribusiness.

Key words: Servicio de información: modelos matemáticos; ejecución de submodelos; agentes informáticos: sistema de información.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La ciencia de la información consiste en administrar y divulgar la información para uso y conveniencia de quienes la necesitan. De ese modo surgió hace varios años la Ciencia del Informar (Informing Science) como resultado de un grupo de trabajo encabezado por Cohen (1999). El objetivo de esta ciencia interdisciplinaria es conjugar esfuerzos de especialistas en diversas áreas del conocimiento para formular y aplicar teorías relacionadas con la comunicación.

Se formularon modelos tipo informador-cliente, donde el primero es el que proporciona la información al segundo. Se contemplan estos tipos de procesos en cualquier ámbito, incluyendo la educación y, de especial interés en esta investigación, el mundo de los negocios. En este contexto, destacan entre los atributos deseables de los procesos de informar la veracidad, puntualidad, credibilidad, que sea verificable, actualizada, interpretable, aplicable y en particular, según Gackovski (2005) que sea relevante para el cliente que la recibe. Sin embargo, este mismo autor señala que en muchas ocasiones, la información debe ser *timely actionable*, lo que significa que el recipiente puede tomar acciones inmediatas resultado de decisiones tomadas en base a la nueva información.

A raíz de lo anterior Bauer (2008) concibió el servicio FLAG como un servicio que informa a sus clientes (*informing service*) de modo que se cumplan los atributos principales enunciados previamente, pero con énfasis en el último: que el cliente reciba la información de tal modo que pueda interpretarla con la premura suficiente para que le sea de utilidad en sus actividades.

El servicio obtiene valores de ciertos aspectos de los negocios de sus clientes. Por ejemplo busca precios de insumos, cotizaciones diversas, datos de clima, con la periodicidad requerida por los interesados, y se los proporciona a los clientes no como “datos” sino los incluye como valores de variables de modelos que formulan los propios clientes.

La ventaja de este modo de comunicar los valores actualizados es que los clientes además del dato mismo, obtienen el impacto que tiene algún cambio de valores sobre sus actividades. El FLAG se concibió para comunidades en el sector agronómico, en particular, los agronegocios. Y como – aparentemente - la preocupación principal de tales empresas es su flujo de efectivo, se concibió el nombre flujo de efectivo en agronegocios, que a su vez resultó en el acrónimo FLAG. Sin embargo de ninguna manera está limitado el servicio a la determinación de flujos de efectivo ni de algún sector de la economía: sirve en cualquier contexto y naturaleza de las variables.

FLAG se basa en 3 componentes fundamentales. Cada uno de sus clientes formula su propio modelo matemático con variables propias (variables del cliente, VCL) que reflejan información de sus actividades y cuyos valores conoce o determina el mismo cliente. También incluye variables que reflejan datos de su entorno, que se denominaron VMR (variables del mundo real). Por ejemplo un cliente indica en su modelo que la venta de su producto será igual al volumen que produce (un dato que él conoce o determina) multiplicado por el precio de venta, que está dado por el mercado, y por ende, varía. FLAG obtiene este precio, por ejemplo cada día, y "ejecuta" el modelo del cliente con el nuevo valor, lo que resulta en que el cliente conoce sus ingresos por este rubro, si el precio de venta fuera el último reportado.

Para proveer este servicio se creó un sistema de información que consta de varios módulos, cada uno de los cuales realiza funciones que en su totalidad permiten informar a los clientes con puntualidad y precisión. Este sistema recibió el mismo nombre que el servicio, y su descripción constituye el objetivo de esta tesis: describir a detalles la creación del sistema y su implementación. En la actualidad existen muchos servicios y productos que ofrecen información actualizada (on-line o no) a sus clientes. Las casas de bolsa permiten a sus clientes tener información actualizada de sus portafolios; servicios meteorológicos informan datos del clima en forma instantánea; se publican en forma instantánea variaciones en cotizaciones de todo tipo de bienes, y la lista de servicios similares es casi inagotable.

Esto a su vez dio lugar al desarrollo de aplicaciones que transforman esta información ya sea en decisiones o en acciones específicas. Se diseñan sistemas que en base al clima y otros datos, encienden o apagan dispositivos de riego; se instalan dispositivos que cambian parámetros en procesos en función de nuevos datos; el mencionado anteriormente cálculo del valor de un portafolio de acciones en función de las cotizaciones, o de alguna conversión entre monedas son sólo algunos de los ejemplos de tales aplicaciones.

En general, cada servicio del tipo mencionado utiliza procesos diseñados *ad hoc*: el consultor financiero elabora una hoja de cálculo a la cual proporciona los valores actualizados de las acciones y se calcula el valor total de la cartera. El FLAG hace algo similar, pero con dos diferencias fundamentales: el sistema es aplicable a cualquier situación, e incorpora varios elementos que sólo están presentes en algunos de los servicios que se ofrecen a los clientes. Estos aspectos se comentarán ampliamente en el resto de este documento, pero entre los principales están que en lugar de un valor de una variable (cotización del peso en el mercado de divisas) usa un vector de 24 valores (llamados períodos),

además de que permite que estos valores sean aleatorios, con distribuciones de probabilidad discretas.

Uno de los objetivos que comparte el FLAG con muchos de los servicios de información a clientes es que al ofrecerlo a muchos clientes, la obtención de los valores actualizados de las variables sirve a muchos de ellos, por lo que se prorratea el costo de mantenerlos actualizados.

Un modelo de un cliente consiste de las variables previamente descritas (VCL y VMR). Algunas de las variables propias se calcularán a partir de otras por medio de fórmulas, que tienen gran flexibilidad pero que no representan retos en cuanto a conocimientos avanzados matemáticos, de modo que serán accesibles a clientes que no cuentan con tales habilidades.

FLAG usa agentes informáticos para obtener valores actualizados de las VMR que ofrece a sus clientes, que en general estarán determinados por las necesidades de éstos. Cuando sea necesario, reacciona a cambios detectados en los valores: ejecuta los modelos (recalcula las fórmulas para obtener los nuevos valores de las variables calculadas); esto lo hace sólo para aquellos modelos que incluyen dichas variables, aunque como se verá, los clientes pueden formular criterios que determinan cuándo desean que se ejecuten sus respectivos modelos.

Finalmente, FLAG ofrece alarmas, que son avisos urgentes a sus clientes cuando, como resultado de la ejecución de sus modelos, algunos indicadores caen fuera de ciertos rangos definidos para ese fin por el cliente.

Inicialmente Bauer (2008) diseño los módulos centrales de FLAG del cliente que, junto con dos alumnos de Maestría, desarrollaron e implementaron partes del sistema en ciertos módulos específicos. Díaz (2009) construyó el módulo de agentes, que obtiene los valores actualizados, y Velázquez (2009) elaboró el módulo de alarmas, que se encarga de determinar la necesidad de enviar una alerta a un cliente e implementa el envío de la alarma por correo electrónico o teléfono celular. El tema de alarmas fue mejorado en el presente trabajo así como también los agentes por Varela (2014).

Sin embargo, esta versión del FLAG no sólo estaba incompleta (algunas de sus funciones estaban implementadas sólo parcialmente) sino que había muchas oportunidades de mejoría. En particular había una situación crítica: la ejecución de los modelos no era suficientemente breve como para ejecutar un gran número de ellos en períodos acotados en cuanto a su duración. Si el FLAG obtuviera valores actualizados de VMR cada minuto, podría resultar en que se tuvieran que ejecutar cientos (o hasta miles) de modelos antes de que surgiera la necesidad de ejecutar el siguiente lote de modelos.

De ese modo se planteó como actividad inicial del proyecto de investigación que aquí se describe tomar una decisión: se completaría y modificaría el sistema anterior, o se construiría un sistema *ab uovo,* es decir, totalmente nuevo. Además de las funcionalidades faltantes o mejorables, se contemplarían las plataformas informáticas utilizadas (el sistema anterior estaba desarrollado en Visual Basic 6.0. Se describe a detalle el proceso que llevó a la decisión de hacer una versión totalmente nueva del FLAG.

De ese modo se llegó a la especificación del objetivo del resto de la investigación:

desarrollar el paquete FLAG con la funcionalidad especificada, con la característica adicional de reducir los tiempos de procesos de la ejecución de modelos; completar y mejorar las interfaces usuario-máquina; eliminación del concepto de sinónimo en la formulación de los modelos de los clientes; completar el módulo de agentes, en especial que permita la obtención simultánea de varios valores; implementar en forma total el “motor” del FLAG, que inicia con la obtención de valores actualizados y finaliza con la ejecución de los modelos afectados y el envió de las alarmas generadas.

Cabe una observación en cuanto al proyecto. Inicialmente se concibió con dos objetivos centrales. El primero, elaborar el sistema FLAG e implementarlo en una aplicación específica: la determinación de riesgos de enfermedades fitosanitarias en el Estado de Morelos. Y segundo, la comunicación a los productores de estos riesgos. Para ello se constituyó un equipo de trabajo formado por el autor de esta tesis, otro alumno de la Maestría (Varela, 2014) y el Dr. Bauer. Por diversas circunstancias se llegó a la conclusión de que no se usaría el FLAG para los riesgos de enfermedades, de modo que el grupo se disolvió; Varela (2014), sin embargo, desarrollaría el módulo de Agentes Informáticos que formaría parte del sistema FLAG total.

Como consecuencia, esta tesis se organizó del siguiente modo. En los materiales se incluyó el módulo de Agentes, puesto que no resultó de la investigación que se describe en esta tesis, a pesar de que sí lo fue en forma parcial en cuanto a la comunicación entre dicho módulo y el resto del motor del FLAG.

A continuación de describe el proceso que resultó en la decisión de elaborar una versión nueva de todo el sistema. Sigue una descripción general de FLAG y la división en los módulos que se detallan en los capítulos siguientes.

Los módulos de actualización de variables, captura de valores, definición de fórmulas y consultas, se rediseñaron totalmente, tanto en cuanto a las interfaces como en el modo de la interacción entre el cliente y el servicio. Se redefinió el proceso que prepara las fórmulas para ejecución, especialmente con la creación

de sinónimos (en la versión anterior los definía el cliente como parte de su modelo) y la implementación de submodelos, ausentes en la versión anterior.

Como se verá, se hicieron varios cambios a la ejecución de un modelo aunque el algoritmo principal es prácticamente el mismo. El módulo de agentes se dividió en dos partes. La primera es sobre la definición de los agentes: los tiempos de envío, valores que debe buscar, etc. Segunda, los envíos, es decir, el momento en que se ejecuta el proceso de los agentes cuando deben de traer valores a cada cierta hora. A diferencia de como se hacía antes, ahora es con multi-hilos para que un envió de un agente no tenga que esperar que termine el anterior.

Los módulos que se agregaron son la captura de criterios de ejecución, que cada cliente debe especificar para las variables del mundo real que está usando.

Se implementó el “motor” del FLAG: obtiene valores actualizados por medio de agentes; aplica los criterios de ejecución a las VMR que cambiaron de valor, y en consecuencia invoca la ejecución de los modelos que deben ser recalculados.

Otro módulo importante es el de la comunicación que se encarga de llevar al sistema FLAG la información de todos los clientes como lo son las bases de datos y los archivos planos. Los archivos tienen información vital para la ejecución de modelos de cada cliente: los valores, las fórmulas y los submodelos.

La ejecución de un submodelo es otro aporte nuevo para esta versión de FLAG, anteriormente se tenía contemplado pero no se programó hasta ahora. Esta parte es muy importante ya que resulta en ejecuciones mucho más eficientes: cuando solo han cambiado los valores de un única VMR en lugar de ejecutar el modelo completo sólo se calculan las fórmulas que son afectadas por ella.

Cuando se ejecuta un tal submodelo, se pueden aprovechar los operandos aditivos de las fórmulas. Esto se explicará a detalle, pero se trata de que un operando aparece como sumando o sustraendo en una fórmula, de modo que en lugar de efectuar todas las operaciones basta aplicar la diferencia detectada en los valores del operando aditivo a los de la fórmula.

1. MARCO TEORICO

1.1. Informing Science

El proyecto FLAG se inscribe en la teoría de Informing Processes. Se trata de crear herramientas para mejorar la comunicación en todos los sentidos, pero en concreto, lograr dos objetivos adicionales a los que se les exige a la información: fundamentales. De ese modo, a las características imprescindibles para que el cliente pueda aprovechar la información recibida,

- Que la información sea relevante;
- Que se elimine en lo posible la complejidad como factor negativo;
- Que la información sea confiable y correcta (en ocasiones se agrega el requisito de que sea también comprobable);
- Que sea interpretable por el cliente;

Se le agregan otros tres aspectos que aumentan la utilidad de la información:

- Que la información llegue “a tiempo”;
- Que la información sea “timely actionable” lo que significa que el recipiente puede usarla de inmediato:
- Que el costo del servicio sea inferior al beneficio obtenido por su uso; en la literatura se usa el término *affordable* para este concepto: preferimos aclarar lo que significa para un cliente, puesto que el vocablo original parecería indicar que “puede pagarlo” (traducción literal) mientras que en la realidad sería mejor decir que “le conviene pagarlo”.

FLAG se concibió precisamente para encarar estos tres objetivos. El uso de modelos formulados por el cliente precisamente para averiguar en forma instantánea el impacto de nuevos datos sobre sus actividades acorta el tiempo entre la recepción de la información y la posibilidad de tomar decisiones basadas en los cambios producidos. Para asegurar que el cliente se entera a tiempo de estos cambios se incluyeron las alarmas (avisos urgentes de cambios significativos en sus actividades) que se envían cuando los nuevos valores de sus indicadores indican que debe conocerlos con rapidez.

El servicio que se concibió tiene como uno de los principales objetivos servir a un conjunto numeroso de clientes, muchos de los cuales requieren conocer los valores actualizados de las mismas variables. El tipo de cambio dólar-peso, el precio de ciertos productos o algún indicador económico puede formar parte de los modelos de muchos clientes. De ese modo el costo de obtener los valores de dichas variables con cierta frecuencia se reparte entre los beneficiarios, que además comparten el costo de la infraestructura necesaria para brindar el servicio.

1.2. Los modelos que se emplean en FLAG

1.2.1. El árbol como estructura de un modelo

En matemáticas y ciencias de la computación, un grafo conexo (del griego grafos: dibujo, imagen) es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto. Una de las estructuras de datos más utilizadas es la llamada árbol, que se define como un grafo conexo sin bucles (también llamados ciclos). Weiss (1993) menciona que un árbol consiste de un nodo distinguido *r*, llamado raíz, con cero o más subárboles T_1, T_2, …, T_k, cuyas raíces están conectadas por un vértice dirigido a *r*; se dice que la raíz de cada subárbol es un hijo de r, y r el padre de cada subraíz.

Para nuestros fines, adoptamos la siguiente terminología: *nodos*; *subnodos*, que son los nodos que están conectados a un nodo; *hoja*, nodo sin subnodos; *vértice*, nodo que no es subnodo de otro; *orden del árbol*, el número máximo de subnodos que puede tener un nodo. Entre un nodo y sus subnodos decimos que hay una relación “padre” e “hijo”.

1.2.2. Los tipos de modelos

El concepto principal que define el servicio FLAG es el modelo del cliente. Sobre éste actúan todos los procesos entre los cuales está la comunicación entre el cliente y FLAG, las consultas, la actualización de variables y la ejecución de las fórmulas de un modelo.

Como se mencionó arriba, la estructura de datos que FLAG usa para sus modelos es el tipo árbol (ver Figura 1). Es conveniente detallar los motivos que hacen que esto sea conveniente: sus ventajas y la comparación con otras que se tomaron en cuenta a la hora de definir el sistema.

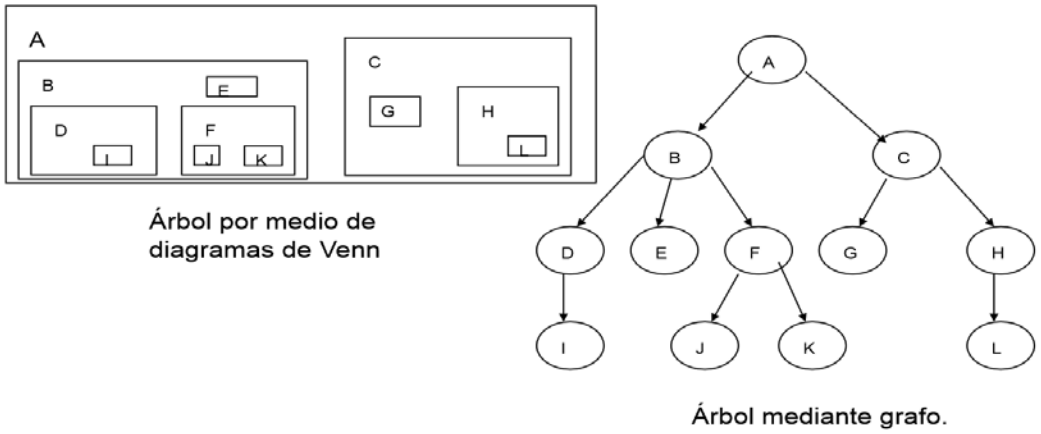


Figura 1. Representaciones de un modelo tipo árbol

La literatura divide a los tipos de árbol en dos: binarios y multicaminos. Lipschutz (2004) menciona que el tipo de árbol binario se caracteriza principalmente por tener un máximo de dos hijos por nodo (de ahí el nombre *binario*). Sin embargo, los árboles multicaminos pueden tener nodos con muchos hijos y lo mismo para éstos. Los árboles binarios se usan con bastante frecuencia ya que hay tipos de problemas en donde las soluciones se pueden reducir a dos opciones, dentro de cada una subdividir en otras dos y así sucesivamente hasta llegar a la solución óptima. Entre los problemas que resuelven están las búsquedas que justamente reciben el nombre de binarias y árboles de decisión.

Por otra parte un árbol multicamino posee un grado *g* mayor a dos, donde cada nodo de información del árbol tiene un máximo de g hijos.

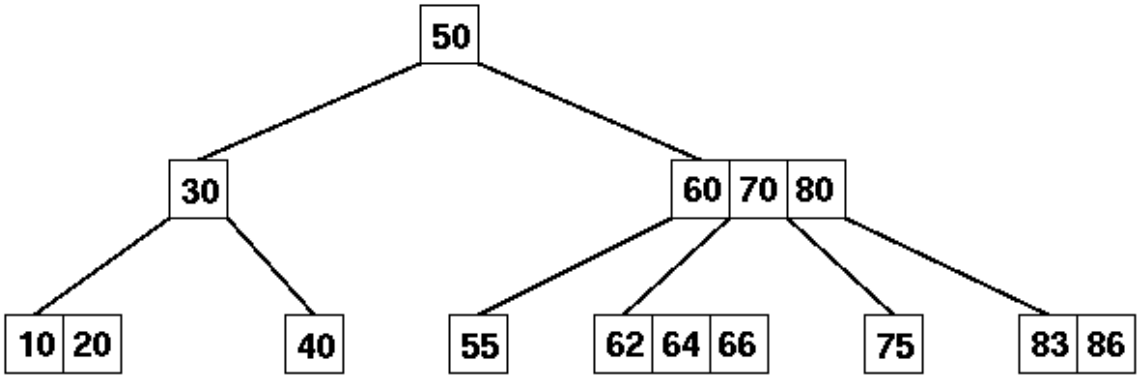


Figura 2. Modelo tipo árbol multicamino

La principal ventaja de este tipo de árboles consiste en que existen más nodos en un mismo nivel que en los árboles binarios con lo que se consigue que, si el árbol es de búsqueda, los accesos a los nodos sean más rápidos (ver Figura 2).

El inconveniente más importante que tienen es la mayor ocupación de memoria- En la implementación más común de árboles de este tipo (los llamados Árboles B) puede ocurrir que una gran parte de nodos no tengan descendientes o al menos no todos los que podrían tener, de modo que se desaprovecha memoria utilizada para almacenar tanto los valores como los diversos apuntadores.

El modelo de árbol usado por el sistema FLAG es de tipo multicaminos ya que cada nodo representa una fórmula que puede contener como operandos otras variables, algunos de ellos a su vez con fórmula. Por una decisión técnica tomada como parte del sistema se decidió que el máximo de operandos en una fórmula, dentro de FLAG sea de 16, lo cual se explicará más adelante. También, el modelo de tipo árbol que será generado, tendrá como máximo 13 subnodos por cada nodo. Lo anterior se entenderá mejor al ver la parte de la estructura de fórmulas, donde se explica su composición y subtotales que puede contener.

1.2.3. Justificación del tipo de modelo usado

La Figura 3 ilustra un ejemplo de un modelo del tipo que contempla el FLAG. Se aprecian los nodos del árbol que están conectados entre sí. Las hojas del árbol son nodos que no tienen subnodos (hijos). Los nodos corresponden a variables que representan algún factor de un negocio o cualquier otro uso del modelo. Los subnodos (hijos) de un nodo representan los operandos de una fórmula mediante la cual se calculan los valores del nodo padre en función de los valores de sus "hijos". De ese modo, los nodos que no tienen subnodos (hojas) no tienen fórmula, y se denominan variables de abajo.

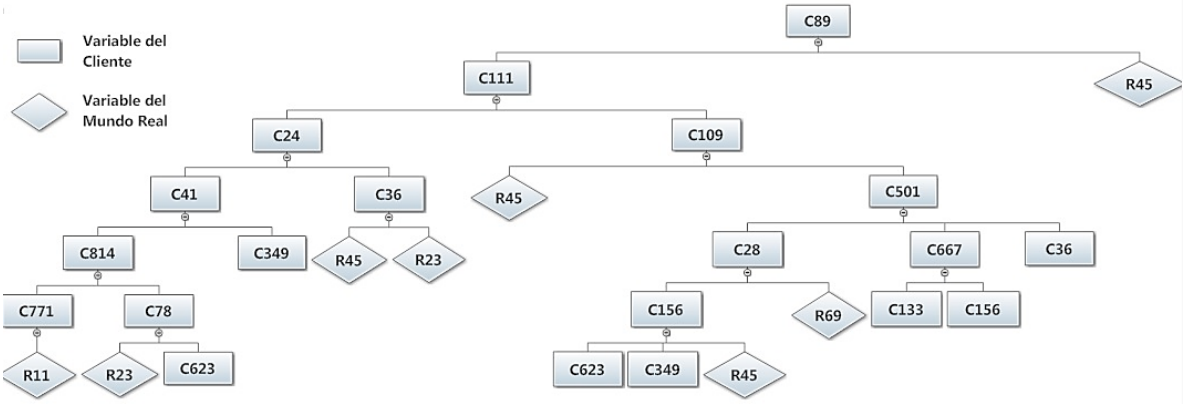


Figura 3. Ejemplo de un modelo de un cliente.

En las hojas de cálculo más populares, entre las cuales mencionaremos a MS-EXCEL, la ejecución de las fórmulas se invoca cada vez que se modifica una celda que la afecta. Sin embargo, hay modos de evitar que esto suceda, indicando que se comience el recálculo con un comando que se invoca en la hoja. En los modelos que se describen aquí, sólo se ejecutará el modelo con una instrucción específica en ese sentido.

Las hojas de cálculo modernas tienen algoritmos sofisticados que les permiten decidir cuáles celdas se deben recalcular como consecuencia del cambio de alguna celda. En ocasiones, los programas deciden que es preferible la ejecución de todo el modelo, cuando la regla de decisión aplicada determine que no vale la pena limitar el recálculo a una parte del modelo. En este trabajo se describe cómo se implementó este mismo tipo de cálculos parciales a los modelos descritos; adelantamos aquí que se hará vía submodelos, que se definen como "el conjunto de fórmulas" que se deben ejecutar para reflejar adecuadamente el cambio de una sola variable.

De la analogía con las hojas de cálculo surge una pregunta: ¿por qué no se usan hojas de cálculo para el servicio FLAG? Entre las causas principales se encuentran:

- Las variables en realidad son vectores de 24 valores cada una;
- Se incluyen variables aleatorias con distribuciones discretas de hasta 4 valores, además de las variables escalares;
- FLAG ejecuta una serie de acciones además de las asociadas al cálculo de fórmulas, y puede ser complicado incluirlas como parte de las hojas de cálculo.

La actualización de los datos que proporciona cada cliente de sus variables “de abajo” se podría hacer directamente sobre una hoja de cálculo, pero sería más difícil para el cliente además de que, al hacerlo, podría destruir alguna relación previamente indicada. Esto es especialmente cierto cuando modifica su modelo, ya sea agregando variables o cambiando alguna fórmula.

Resumiendo, se tomó la decisión – como parte del diseño del servicio FLAG – que no se basaría en hojas de cálculo sino en estructuras propias del sistema desarrollado para implementar el servicio. En particular, se almacenan las fórmulas en estructuras separadas de las usadas para los valores de las variables del modelo.

Para cada modelo, es decir, para cada cliente, se almacenan los valores de todas sus variables en un archivo destinado a tal efecto. Hay un archivo similar para las VMR, pero éste será único. La ejecución de una fórmula resulta en la actualización de los valores de la variable calculada en el archivo de valores. Para cada operando de una fórmula, se usan estos valores, es decir, se cargan a memoria los del registro correspondiente.

1.2.4. Las fórmulas de los modelos

La definición matemática, según la RAE, de “fórmula” es: ecuación o regla que relaciona objetos matemáticos o cantidades. Un cliente define sus fórmulas: especifica la variable (resultado) y los objetos (operandos) que relaciona mediante operaciones. Los operandos (variables), que figuran como hijos, a su vez pueden tener fórmula (ver Figura 3)

Procede un comentario sobre el orden de las operaciones y los paréntesis. Muchas personas tienen dificultades para especificar cálculos usando paréntesis, y otros de hecho no los usan pero definen mal las fórmulas. Por ejemplo, a pesar de que desean indicar (A + B + C) * D, simplemente le quitan el paréntesis, lo que naturalmente resulta en un cálculo diferente al deseado.

Se contempló el uso de notación postfija para definir las fórmulas. Esto se descartó de inmediato, pues a pesar de que hay personas que han usado calculadoras científicas y están familiarizadas con este tipo de indicaciones, la mayoría de los usuarios potenciales no podría hacerlo.

De ese modo se usó otro método para especificar fórmulas en FLAG: para sustituir los paréntesis, se agrupan las operaciones en (hasta) 4 subtotales. El último subtotal proporciona el valor de la variable calculada con la fórmula. De ese modo, una formula consiste de operaciones que involucran operandos, que pueden ser variables o subtotales calculados anteriormente. Los siguientes ejemplos se basan en el ejemplo de modelo ilustrado en la Figura 3.

Fórmula de la VCL156. En lugar de escribir

VCL156 = (VCL623 + VCL349) * VMR045,

se especifica

Subtotal (1): VCL623 + VCL349
Subtotal (2) = subtotal (1) * VMR045

Hay dos aspectos en cuanto al orden en el que se ejecutan las fórmulas. Por un lado, un operando calculado (es decir, corresponde a una variable con fórmula) se debe calcular antes de ser usado como como operando. Adicionalmente, se debe garantizar que no habrá bucles (una variable interviene en una fórmula de otra variable que directa o indirectamente interviene en la del operando). Para evitar esto se implementó el concepto de nivel, que se explica más adelante.

1.3. Probabilidades (Discretas)

FLAG permite usar variables aleatorias en los modelos. Como resultado de las decisiones tomadas durante el diseño del paquete, estas variables se limitaron a aquéllas que tienen:

- Una distribución discreta de probabilidad
- Y un máximo de 4 valores.

En los modelos, podrá haber operaciones entre variables aleatorias y otras entre variables aleatorias y escalares. Por ejemplo, sea V_Esc(4) una variable escalar, y V_Ale(1) y V_Ale(2) variables aleatorias. Entonces podría haber fórmulas como éstas:

V_Ale (15) = V_Ale (1) + V_Ale (2)

V_Ale (1) = V_Ale (1) * V_Esc (4)

Pero también se podrán usar funciones con argumentos aleatorios, como

V_Esc (20) = media (V_Ale (1)).

En el resto de esta sección se explican las variables aleatorias y cómo se efectúan las operaciones que involucran alguna variable aleatoria en los modelos del FLAG.

1.3.1. Introducción a las distribuciones discretas de probabilidad

Observación: el material de este capítulo referente a las distribuciones se puede encontrar en cualquier libro de texto de Probabilidades o de Introducción a la Estadística.

Sea una variable aleatoria X que toma los valores (ver Tabla 1)

Tabla 1. Ejemplo de variable de distribución discreta

Valor de X	**1**	**2**	**3**
Probabilidad	1/3	1/6	½

Observaciones:

- La suma de las probabilidades debe ser 1 (de lo contrario, no sería una distribución de probabilidad);
- Tiene que haber un número FINITO de valores que puede tomar la variable aleatoria (de lo contrario no sería discreta).

Las distribuciones discretas más comunes son: la uniforme, la binomial, la exponencial discreta, la multinomial. En nuestro caso, se trata de otra distribución (es decir, ninguna de las mencionadas).

Se especifica en forma individual la probabilidad de cada uno de los valores que puede tomar la variable aleatoria.

Otro ejemplo de una variable discreta (ver Tabla 2): sea X una variable que representa el "rendimiento por hectárea" de un cultivo

Tabla 2. Ejemplo 2 de variable de distribución discreta

Valor de X	**20**	**30**	**40**
Probabilidad	0.2	0.6	0.2

donde los valores indicados para x realmente representan intervalos:

Tabla 3. Intervalos de los valores de X

Rendimiento entre	15 – 25	25.001 – 35	Más de 35
Valor asignado a la variable	20	30	40

Observe que la asignación de los valores es arbitraria, la hace el que define la variable (ver Tabla 3), o se puede usar alguna regla, por ejemplo:

> *Usar el punto medio del intervalo (promedio de los puntos extremos). En nuestro caso, esto no funcionaría para el último valor.*

1.3.2. Densidades limitadas a 4 valores (decisión del diseño)

Se incluyeron variables aleatorias como posibilidad en los modelos que implementa el FLAG para representar situaciones reales. Para una variable puede haber más de una fuente de información, lo que hace que haya que elegir cuál de ellos es el "verdadero". Pero cuando hay más de un servicio *serio* que proporciona información y ésta difiere entre ellos, se puede decir que se trata de un valor aleatorio y que los métodos de obtención del valor más probable difieren.

El servicio FLAG puede obtener información sobre el éxito de los pronósticos formulados por diversas fuentes de información respecto a un mismo dato. De ese modo, puede informar al cliente que hay 3 valores, a cada uno de los que le asigna una medida de credibilidad, lo que constituye una probabilidad.

Cuando se combinan variables de este tipo en una operación, aumenta el número de valores que puede tomar el resultado. Proporcionarle a un cliente un conjunto de 9 valores con sus respectivas probabilidades no le será de gran utilidad. Por otra parte, si el resultado se limita a 2 valores (por ejemplo) se puede perder la variabilidad que precisamente motivó la inclusión de las variables aleatorias en los modelos.

De ese modo, Bauer (2008) definió que un máximo de cuatro valores es suficiente para la credibilidad de los valores probables de una variable aleatoria discreta. El uso de más de cuatro no solo no ayudaría en el impacto de los cambios sino que complicaría el cálculo de fórmulas, cuestión que se reflejaría en tiempo de cómputo, diseño e implementación. Hasta ahora no se han observado desventajas considerables ni deficiencias conceptuales en el uso de cuatro valores; de ocurrir algo así, el diseño se modificaría sin mayor queja, ajustando el sistema al mejor desarrollo posible del servicio que proporcionará FLAG.

1.3.3. Operaciones entre variables aleatorias discretas

Hay 3 tipos de operaciones en las que se utilizan las variables aleatorias:

- Entre 2 variables aleatorias
- Entre 1 variable aleatoria y otra escalar
- Funciones (escalares) de variables aleatorias.

Se usará la siguiente notación:

Variable aleatoria X1; N1 = número de valores posibles de la variable aleatoria

Análogamente, en la Tabla 4 se puede ver X2 y N2.

Tabla 4. Dos distribuciones de probabilidad

Valor \ número	***1***	***2***	***3***
Valor de X1 (N1 = 3)	X1(1)	X1(2)	X1(3)
Probabilidad	P1(1)	P1(2)	P1(3)
Valor de X2 (N2 = 2)	X2(1)	X2(2)	
Probabilidad	P2(1)	P2(2)	

IMPORTANTE: en todos los casos, se suponen que las variables que intervienen en las operaciones son INDEPENDIENTES.

1.3.3.1. Operaciones entre variables aleatorias

En probabilidades se usa el término “convolución” de las distribuciones para efectuar operaciones entre dos funciones aleatorias, ya sean continuas o discretas. Laha (1979) presenta el *Teorema de Convolución* de la siguiente forma:

$$F = F_1 * F_2 \quad \leftrightarrow \quad \varphi = \varphi_1 \varphi_2$$

donde F, F_1 y F_2 son tres funciones con las características φ, φ_1y φ_2 respectivamente. En matemáticas y, en particular, análisis funcional, una convolución es un operador matemático que transforma dos funciones F_1 y F_2 en una tercera función que, en cierto sentido, representa la magnitud en la que se superponen F_1 y una versión trasladada e invertida de F_2. Una convolución es un tipo muy general de media móvil.

En teoría de la probabilidad, la distribución de probabilidad de la suma de dos variables aleatorias independientes es la convolución de cada una de sus distribuciones de probabilidad.

Tomando lo anterior como premisa, se efectúan las operaciones entre funciones aleatorias discretas. Como las “fórmulas” resultarían difíciles de leer, se usaron valores y probabilidades al siguiente ejemplo que usaremos para ilustrar todas las operaciones.

Variable aleatoria X1 y N1 = número de valores posibles de la variable aleatoria

Tabla 5. Ejemplo de dos distribuciones de probabilidad discretas y sus valores

Valor \ Número	***1***	***2***	***3***
Valor de X1	4	5	5
Probabilidad	0.25	0.5	0.25
Valor de X2	2	3	
Probabilidad	0.5	0.5	

Sean 2 variables aleatorias X1 y X2. Sean N1 = 3 y N2 = 2 (ver Tabla 5)

SUMA: X3 = X1 + X2 (X3 es otra variable aleatoria con distribución p3)

Comencemos con un concepto intuitivo. El resultado de la suma de un valor de X1 y un valor de X2 será la suma de esos valores. De modo que el dominio de la variable X3 será el obtenido de las sumas de todas las combinaciones de valores de X1 y X2.

Observe que si uno calcula las sumas de todos los pares, puede haber SUMAS REPETIDAS.

Se calculan las probabilidades de cada uno de las combinaciones diferentes de la suma. Para ello se calculan las probabilidades de X1(i) + X2(j) para i = 1, 2, 3 y j = 1, 2. Como las variables son INDEPENDIENTES, la probabilidad de cada una se obtiene como P(suma) = p1(i) * p2 (j). El resultado se muestra en la Tabla 6.

Tabla 6. Resultado parcial de la suma de dos probabilidades discretas

X1	*p1*	*X2*	*p2*	*Suma*	*Probabilidad*
4	¼	2	½	6	1/8
4	¼	3	½	7	1/8
5	½	2	½	7	¼
5	½	3	½	8	¼
6	¼	2	½	8	1/8
6	¼	3	½	9	1/8

Ahora se combinan los resultados repetidos indicados por el mismo valor (sumamos las probabilidades de los eventos con el mismo resultado) y obtenemos

Los valores y probabilidades mostrados en la Tabla 7.

Tabla 7. Probabilidad obtenida de la asociación de resultados de la suma parcial

Suma	*Se suman probabilidades*	*Probabilidad*
6	1/8	1/8
7	1/8 + ¼	3/8
8	¼ + 1/8	3/8
9	1/8	1/8

Como no se manejan variables de más de 4 valores, surge un problema: ¿qué pasa si (aún después de juntar los “repetidos”) quedan más de 4 valores? A continuación se explica cómo el FLAG resuelve esta situación.

Construcción de una variable de (máximo) 4 valores a partir de otra que tiene más de 4 valores

Observe que el método utilizado es una decisión tomada para el FLAG, y no constituye algún método aceptado o recomendado por la literatura. De hecho, es **UNA** manera de construir esta nueva distribución, y no **LA** manera de hacerlo. Bauer (2008) desarrolló esta metodología tomando en cuenta el concepto de convolución, mencionado antes, sin embargo esta forma está hecha *ad hoc* para

el funcionamiento del sistema FLAG; la responsabilidad de esta decisión recae en el autor de esta metodología.

Los valores llegan ordenados por valor. El algoritmo seleccionado se basa en "combinar valores parecidos (en valor) de modo que son "vecinos". Para ello seleccionamos el valor de menor probabilidad y, de sus vecinos, una vez más el que tiene la menor probabilidad. Reemplazamos estos dos valores por UN VALOR (el promedio de los dos), y le asignamos la probabilidad obtenida sumando la de los dos valores que se combinaron.

Esta operación se repite hasta que el número de valores sea 4 (recordemos que se utiliza sólo cuando hay más de 4 valores, y se reduce en 1 este número en cada iteración).

Las restantes operaciones entre variables aleatorias

Para la diferencia, el producto y la división de variables aleatorias se procede del mismo modo. Se calculan todos los resultados posibles y luego se acumulan las probabilidades de cada una de las combinaciones de valores de los operandos.

Operaciones entre más de 2 variables aleatorias

Aquí se tomó una decisión: el cálculo se hace "por pares", es decir, se efectúa la operación entre los primeros 2 operandos. Si procede, se construye la distribución con máximo 4 valores; a ésta se le aplica el 3er operando, etc.

La alternativa contemplada (y desechada) era calcular todos los valores posibles de las operaciones entre todos los operandos; luego se acumulan, y finalmente se construye una distribución de máximo 4 valores (si hubiera más). Esto, en términos de probabilidades, es superior al método adoptado, pero se descartó por el cómputo implicado y porque, al construir la distribución final, de todos modos se obtiene una aproximación.

1.3.3.2. Operaciones entre 1 variable aleatoria y otra escalar

Se obtiene una nueva variable aleatoria con el mismo número de valores. Cada uno se obtiene aplicando la operación a ese valor, y las probabilidades son las mismas.

1.3.3.3. Estadísticas usadas en los modelos (M, Me, Mo, Va, DE)

El FLAG usa (e implementa) las siguientes operaciones:

- Media
- Moda
- Mediana

- Varianza
- Desviación media estándar

A pesar de que son muy conocidas, explicaremos estas funciones. Para ello usaremos la notación introducida anteriormente para la variable aleatoria X1.

Media (X1) = x1 (1). p1(1) + x1 (2) . p1(2) + x1 (3) . p1(3)

O en general

$$\sum_{i=1}^{N} x_i p_i$$

Es decir, se calcula el momento de orden 1 (la esperanza matemática) de la distribución.

Moda = X(j) tal que p(j) = max (p(i)), i = 1 to N

Mediana: x(j) donde j = máximo h tal que

$$\sum_{k=1}^{h} p(k) < 0.5$$

Varianza: es el segundo momento respecto de la media. Sea μ = Media (x1)

$$Var(X) = \sum_{i=1}^{n} p_i \cdot (x_i - \mu)^2 = \sum_{i=1}^{n} (p_i \cdot xi^{\,2}) - \mu^2$$

Desviación estándar = $\sqrt{Var(X)}$

1.4. Estructuras y Análisis de Bases de Datos

1.4.1. Teoría de Bases de Datos

Un sistema manejador de bases de datos (DBMS, por sus siglas en inglés) consiste en: una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos. Deco (2013) detalla que están diseñados para gestionar grandes bloques de información:

- Definición de estructuras para el almacenamiento de información
- Provee mecanismos para la gestión de información
- Mantenimiento de la seguridad de la información almacenada (caídas del sistema, accesos no autorizados)
- Control de concurrencia: Provee mecanismos que eviten posibles resultados anómalos en datos compartidos por varios usuarios

También se encargan de evaluar la redundancia e inconsistencia de datos, es decir, los archivos y los programas de aplicación son creados por distintos

programadores en distintos momentos, por lo cual es probable que los archivos tengan diferentes formatos y programas, por lo que pueden estar duplicados en varios sitios.

Un objetivo importante del DBMS es dar a los usuarios una visión abstracta de los datos, y extraerlos en forma eficiente. Para entender lo anterior Deco (2013) explica tres niveles de los DBMS:

- *Nivel físico (o interno)*: describe cómo se almacenan los datos. Se describen en detalle las estructuras de datos complejas.
- *Nivel conceptual (usado por los administradores de BD)*: describe qué datos son realmente almacenados en la BD y las relaciones que existen entre los datos.
- *Nivel de visión (o externo) (usado por el usuario):* describe sólo parte de la BD completa.

Un modelo de datos es una colección de herramientas conceptuales para describir datos, relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia, todo lo cual permite describir la estructura de una BD.

El modelo entidad-relación (E-R) se basa en una percepción de un mundo real que consiste en una colección de objetos básicos llamados entidades y relaciones entre estos objetos.

- Entidad: objeto distinguible de otros por medio de un conjunto específico de atributos
    - Ej.: número y saldo describen una cuenta de un banco.
- Relación: asociación entre entidades.
    - Ej.: la relación cuenta – cliente asocia a un cliente con cada una de las cuentas que tiene.

Dos conceptos muy importantes son instancia y esquema. Instancia de la BD es la colección de información almacenada en la BD en un determinado momento en el tiempo. Esquema de la BD es el diseño global de la BD.

Los sistemas de BD tienen varios esquemas: un esquema físico (nivel de abstracción más bajo), un esquema conceptual (nivel intermedio) y uno o más subesquemas (nivel más alto - vistas)

Otro concepto es la independencia de datos, que se refiere a la capacidad de modificar una definición de un esquema en un nivel sin afectar la definición de un esquema superior siguiente.

- La Independencia física es la capacidad de modificar el esquema físico sin tener volver a escribir los programas de aplicación.
- Independencia lógica es la capacidad de modificar el esquema conceptual (alterar la estructura lógica de la BD), sin tener que volver a escribir los programas de aplicación. Ej.: añadir un nuevo campo

1.4.2. Almacenamiento y Manejo de Información en una BD

La información en una base de datos tiene especificaciones para su almacenamiento y manejo. Lenguaje de definición de datos (DDL) es un conjunto de definiciones que especifica un esquema de BD. Por ejemplo un diccionario de datos (catálogo o directorio) que es un archivo que almacena un conjunto de tablas, contiene metadatos, y se consulta antes de leer o modificar los datos reales en el sistema de BD.

Otro aspecto importante relacionado con los datos es su manipulación, es decir, la recuperación, inserción, supresión y modificación de datos almacenados en la BD. Para esto se concibió el lenguaje de manipulación de datos (DML) que permite a los usuarios a acceder o manipular datos.

- Procedimentales: el usuario debe especificar qué datos se necesitan y cómo obtenerlos.
- No procedimentales: el usuario debe especificar qué datos se necesitan sin especificar cómo obtenerlos.

Para la recuperación de datos existe lo que se llama “consulta”, que es una sentencia que solicita la recuperación de información.

Gestor de BD

Es un módulo de programa que proporciona el interfaz entre los datos de bajo nivel almacenados en la BD y los programas de aplicación y consultas hechos al sistema. El gestor de BD es responsable de las siguientes tareas:

- Definición y manipulación de datos
  - El DBMS traduce las distintas sentencias DDL y DML a comandos del sistema de archivos de bajo nivel.
  - Es responsable del verdadero almacenamiento, recuperación y actualización de los datos en la BD.
- Implantación de la integridad.
  - Los valores de los datos que se almacenan en la BD deben satisfacer ciertos tipos de restricciones de consistencia. Ejemplo: saldo de una cuenta bancaria no negativo.
  - El DBA debe especificar explícitamente estas restricciones.
  - El DBMS controla si se violan estas restricciones y toma las acciones apropiadas.
- Implantación de la seguridad
  - No todos los usuarios de la BD necesitan tener acceso a todo su contenido.
  - El DBMS hace que se cumplan los requisitos de seguridad definidos por el DBA.
- Copia de seguridad y recuperación

    - Es responsabilidad del DBMS detectar fallos (rotura de disco, corte de energía, etc.) y
    - restaurar la BD al estado que existía antes de ocurrir el fallo.
- Control de concurrencia
    - Cuando varios usuarios actualizan la BD concurrentemente, es posible que no se conserve la consistencia de los datos.
    - El DBMS controla la interacción entre los usuarios concurrentes

Administrador de la BD

El administrador de datos (DA) es la persona que toma las decisiones estratégicas y de política con respecto a la información de la empresa. El DBA (administrador de la base de datos) proporciona el apoyo técnico necesario para poner en práctica las decisiones del DA. Está encargado del control general del sistema en el nivel técnico. Las funciones del DBA incluyen:

- Definición del esquema conceptual
    - Realizado el diseño lógico de la BD, el esquema conceptual se crea escribiendo un conjunto de definiciones DDL.
    - El DBMS traduce estas sentencias a un conjunto de tablas.
- Definición del esquema interno. (Definición de la estructura de almacenamiento y del método de acceso).
    - El DBA debe decidir cómo se representará la información en la BD almacenada (diseño físico). Ejemplo: disco donde almacenar datos, definición de índices, etc.
- Modificación del esquema y de la ubicación física. El DBA debe supervisar el desempeño y realizar los ajustes apropiados cuando cambien los requerimientos
- Concesión de autorización para el acceso a los datos. Permite al DBA regular qué partes de la BD van a poder ser accedidas por varios usuarios
- Especificación de las restricciones de integridad.
    - Los valores de los datos que se almacenan en la BD deben satisfacer ciertos tipos de restricciones de consistencia. Ejemplo: el saldo de una cuenta bancaria no debe ser negativo.
    - El DBA debe especificar explícitamente estas restricciones
- Definición de procedimientos de respaldo y recuperación

Usuarios de la BD

Existen distintos tipos de usuarios que pueden interactuar con una base de datos. A continuación se menciona cada uno con su respectiva función en el sistema:

- Programadores de aplicaciones
    - Interaccionan con el sistema por medio de llamadas en DML incorporadas en un programa escrito en un lenguaje principal.
    - Un precompilador de DML, convierte las sentencias DML a llamadas normales a procedimientos en el lenguaje principal.
- Usuarios que escriben sus preguntas con un lenguaje de consulta de BD.

- Usuarios que interactúan con el sistema invocando a uno de los programas de aplicación existentes.

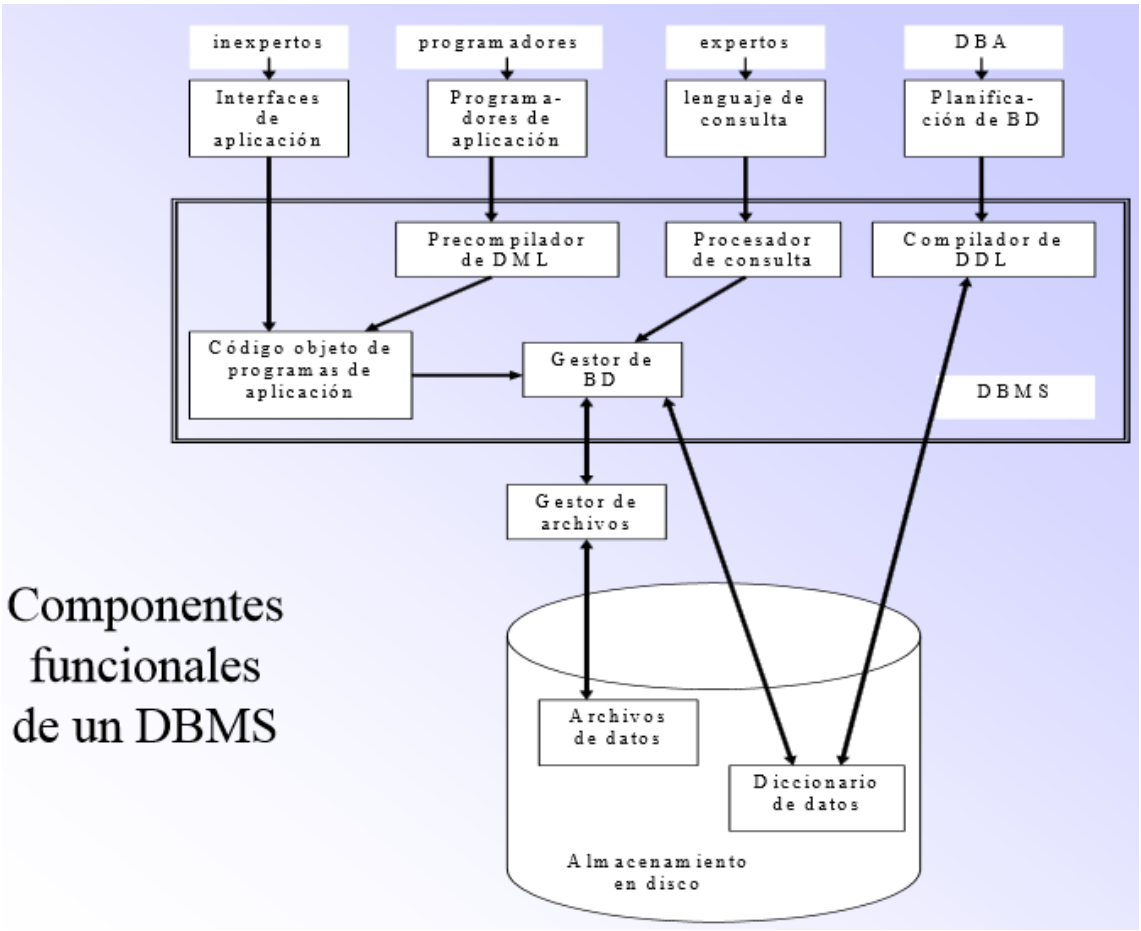


Figura 4. Esquema de los componentes de un DBMS. Imagen: Deco (2014)

La Figura 4 ilustra los componentes de un DBMS que se encargan de implementar todas las funcionalidades descritas.

1.4.3. Ventajas

Las ventajas principales del uso de sistemas basados en el enfoque de bases de datos es que posibilitan:

- Disminuir la redundancia;
- Evitar las inconsistencias;
- Compartir datos:
    - Aplicaciones ya existentes pueden compartir información de la BD.
    - Se pueden desarrollar aplicaciones nuevas para trabajar con los mismos datos almacenados.

- Hacer cumplir las normas: el DBA (Data Base Admin) puede garantizar la aplicación de normas para la representación de los datos;
- Aplicar restricciones de seguridad.
    - El DBA puede asegurar que el acceso a la BD sea sólo a través de los canales apropiados
    - y, por tanto, puede definir las verificaciones de seguridad por realizar cuando se intente acceder a información restringida;
- Mantener la integridad: el DBA puede definir verificaciones de integridad que se aplican en toda operación de actualización de datos.

1.4.4. Bases de Datos en el Contexto de FLAG

El sistema FLAG, debido a su extensión y complejidad, hace uso de bases de datos para el almacenamiento de información. Se adoptó este tipo de estructura puesto que permitiría administrar los datos de cada cliente en forma eficaz sin complicar en demasía la operación del servicio. Por otro lado la desventaja que tienen las bases de datos sobre los archivos planos es que generan un *overhead* considerable. Esta circunstancia no afecta los procesos del FLAG; puesto que en el motor del FLAG no se usan bases de datos, sino solamente archivos planos en los que se almacenan las fórmulas y los valores de las variables. La excepción es en que algunos procesos, al inicio de una sesión, cargan ciertos datos residentes en bases de datos a memoria.

El sistema FLAG guardará datos tales como los usuarios del mismo, los clientes, datos de los clientes, variables del mundo real (incluyendo sus atributos), agentes informáticos, variables de clientes (más atributos), criterios de alarma, criterios de ejecución de sus VMR y consultas catalogadas. La mayor parte de la información del sistema se concentrará en bases de datos. En cambio, hay otro tipo de información que no será almacenado en bases de datos, tal es las fórmulas (del cliente y del mundo real), valores de las variables y los submodelos a ser usados en los criterios de ejecución. Las razones se explicarán más adelante, el lector se dará cuenta que está bien justificada esta decisión.

1.5. Agentes

1.5.1. Definición de agentes

En la historia y más allá de la Inteligencia Artificial, menciona Berrocal (2003), el término agente ha sido usado con dos acepciones. Primero, a partir de Aristóteles y hasta nuestros días, en filosofía el término agente se ha referido a una entidad que actúa con un propósito dentro de un contexto social. Segundo, la noción legal de agente, como la persona que actúa en beneficio de otra con un propósito específico, bajo la delegación limitada de autoridad y responsabilidad, estaba ya presente en el derecho romano y ha sido ampliamente utilizada en economía.

Lara (2004), define un agente como una entidad autónoma capaz de almacenar conocimiento sobre sí misma y sobre su entorno, con unos objetivos y capacidad. Asimismo, un agente inteligente es un programa que basándose en su propio conocimiento, realiza un conjunto de operaciones para satisfacer las necesidades de un usuario o de otro programa, bien por iniciativa propia o porque alguno de estos se lo requiere.

En el contexto de la computación, el concepto de agente se consolida como una solución a las demandas actuales: ubicuidad, interconexión, inteligencia, delegación y homocentrismo. Los agentes inteligentes emergen como la herramienta para delegar adecuadamente nuestro trabajo y abordar esta problemática desde una perspectiva más familiar para usuarios, programadores y diseñadores.

1.5.2. Tipos de agentes inteligentes

Los agentes inteligentes son programas definidos por diversos autores, en ocasiones de manera lo suficientemente poco precisa como para que se produzca discusión acerca de si tal o cual sistema es o no un agente. Las clasificaciones que podemos hacer de los agentes son variadas y dependen de las diferentes características que se tengan en cuenta. Berrocal (2003) menciona la clasificación de la Figura 5.

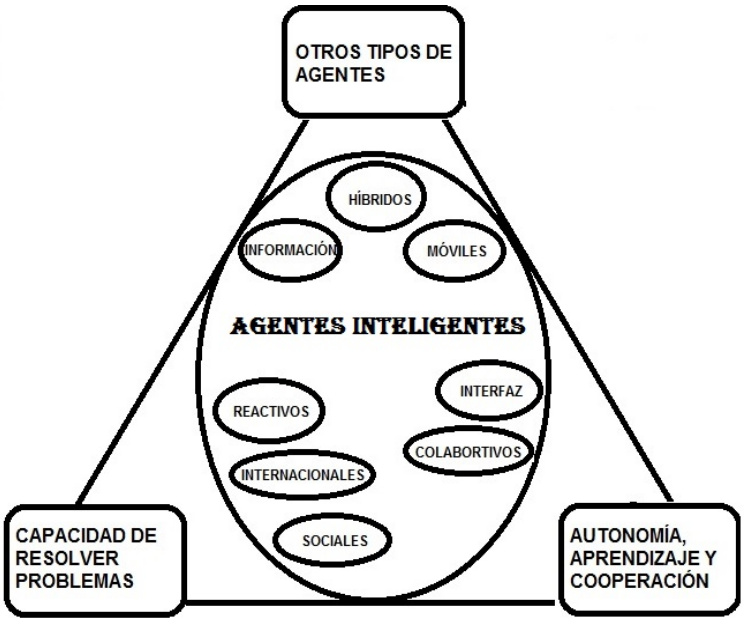


Figura 5. Tipos de agentes inteligentes

Los agentes intentan ofrecer una solución a este tipo de situaciones; esto es importante pues en muchos casos son de vital importancia en la recuperación de

información por la calidad de sus contenidos. Berrocal (2003) menciona que las características típicas de un agente inteligente son:

Autonomía. Los agentes deben trabajar sin supervisión humana, al contrario de los programas que operan en base a interfaces de manipulación directa por parte del usuario. Así, una vez fijadas las condiciones y restricciones necesarias por parte del usuario, se espera que el agente intente cubrir o conseguir sus objetivos, dejando ocultos los detalles para dicho usuario.

Cooperación. Un agente debería ser capaz de colaborar con otros agentes, intercambiando información y resultados de acciones propias. La negociación puede hacerse con agentes que persigan los mismos objetivos (de manera que, por ejemplo, objetivos ya logrados por un agente podrían ser cedidos a otro agente que persigue lo mismo), o con agentes de objetivos diferentes, pero necesarios o al menos útiles para lograr las metas del primero. La capacidad de cooperación requiere disponer de algún mecanismo que permita la negociación entre agentes, aun cuando éstos sean heterogéneos. Se han propuesto diversos estándares, siendo probablemente el más difundido el conocido como KQML.

Comunicación. Esto no sólo implica la simple capacidad de comunicarse con el usuario, sino también la necesidad de tener conocimiento sobre el mundo o dominio sobre el cual opera el agente. Habitualmente este conocimiento se implementa mediante ontologías y reglas de inferencia.

Reactividad. Un agente debería poder responder ante eventos, tomando sus propias decisiones, incluso modificando su manera de operar, siempre con vistas a la consecución de sus metas.

Adaptatividad. Un agente debería poder aprender de experiencias pasadas (y de la experiencia de otros agentes), así como de las reacciones del usuario ante resultados previos. Esto está directamente relacionado con el aprendizaje de máquina o aprendizaje automático

Las capacidades de los tipos de agentes varían dependiendo de los objetivos para los cuales fueron creados. Lara (2004) explica una serie de tipologías de herramientas de segunda generación:

Clientes z39.50. Permiten la consulta simultánea de un elevado número de servidores, mediante un único protocolo, es decir, una única interfaz y lenguaje de interrogación. Son especialmente útilizados en la recuperación de la información que se encuentra en la llamada “Internet invisible”, información que no es indizada por los motores de búsqueda – por ejemplo, las bases de datos.

Volcadores. Permiten volcar automáticamente una copia idéntica de sedes, directorios y documentos, manteniendo su estructura y sus elementos – incluso los enlaces - creando así un archivo offline. Se puede programar la hora del volcado,

reduciendo considerablemente el tiempo y el coste, y permite activar el vuelco de diferentes tipos especiales de documentos (.html, .doc, .pdf, .gif).

Mutibuscadores o metabuscadores. Permiten realizar la recuperación de la información en varios motores de búsqueda simultáneamente. A diferencia de los multibuscadores de primera generación, la mayoría de las tareas pueden automatizarse y son muy flexibles en su configuración: traducen expresiones a lenguaje natural, envían los perfiles a varios motores de búsqueda y procesan los resultados, eliminando los duplicados, y ordenando los contenidos según criterios y formatos definibles.

Trazadores. Permiten la búsqueda en las páginas enlazadas desde una página web determinada o desde una lista de resultados de un buscador. Desde esta primera sede, llamada “semilla”, y aprovechando la naturaleza hipertextual de Internet, se van comprobándo las páginas que se encuentran enlazadas según una serie de criterios de pertinencia, y así sucesivamente hasta un nivel prefijado. Aunque generan mucho ruido y es una técnica lenta, permiten recuperar información que los buscadores no pueden encontrar.

Indizadores. Permiten indizar y resumir automáticamente diferentes páginas web, y exportar los resultados en diferentes formatos reutilizables por editores web.

Mapeadores. Describen íntegramente una sede, detallando cada fichero y directorio, y proporcionando un mapa de contenidos. Permiten obtener datos numéricos que ayudan a evaluar dichos contenidos y establecer una comparativa entre diferentes sedes web – en base a valores como el tamaño, la densidad hipermedia de la sede, su estructura de niveles, la tipología de enlaces, etc.

El uso de agentes en la esta investigación se basa en el envío de programas para la búsqueda de información de ciertas variables que están en constante cambio. Más adelante se detallarán los componentes de los mismos y la forma en que serán usados para beneficio de clientes del sistema FLAG.

2. LOS MÓDULOS DEL FLAG

2.1. Introducción

FLAG es un conjunto de infraestructura, módulos, procesos e interfaces que interactúan de manera distinta para formar un proceso general en el que se involucra al cliente así como al programa general de FLAG.

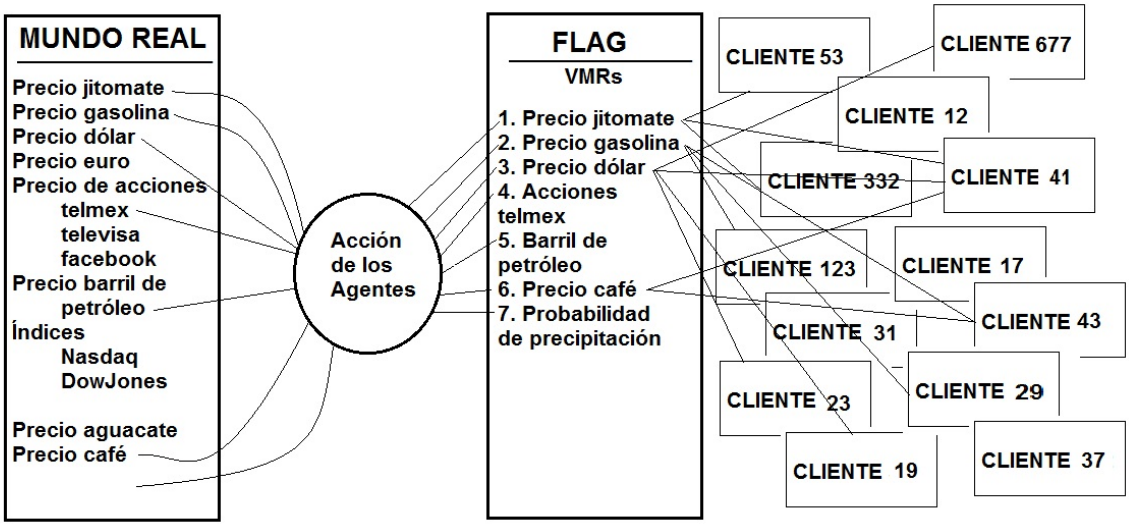
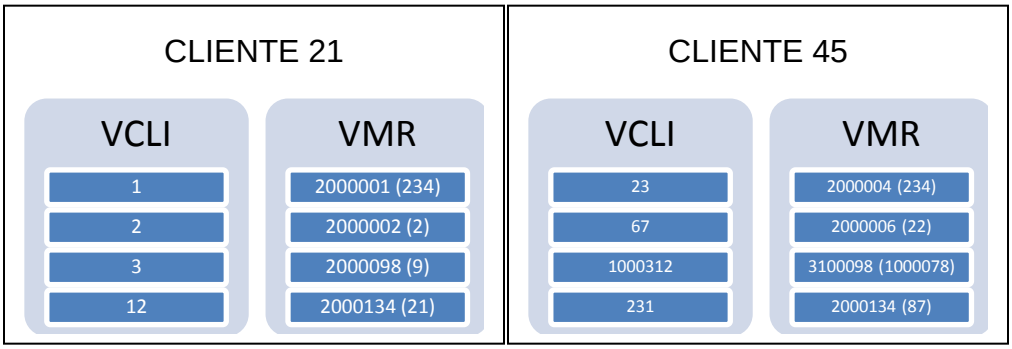


Figura 6. Esquema de funcionamiento del sistema FLAG

El servicio que proporciona el FLAG es que por, medio de los agentes de información, consigue los valores de ciertas variables que están en el mundo real (es decir, el entorno de negocios de sus clientes). Se almacenan en estructuras específicas en FLAG y se ponen a disposición de los clientes que cuentan con este servicio. Cada cliente crea su propio modelo a su conveniencia: puede incluir variables propias más otras que son del mundo real, como se ve en la Figura 6.



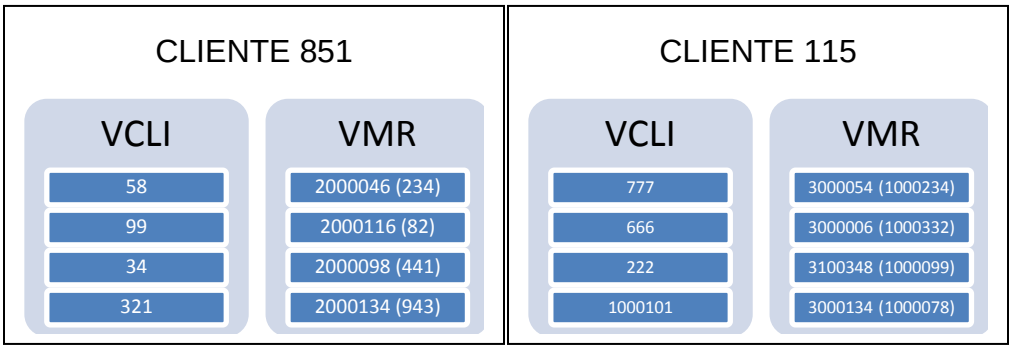


Figura 7. Ejemplos de clientes con variables de cliente y variables del mundo real

En la Figura 7 e pueden ver las variables que contiene cada modelo de un cliente, distintas unas de otras con respecto a los demás clientes, pero también VMR que aparecen en varios de los modelos.

El número que se asigna a las VMR en cada modelo es un consecutivo dentro de ese modelo específico, pero se refieren a un número original que viene de FLAG (entre paréntesis en la imagen).

Lo anterior da una idea general de lo que pretende hacer el sistema FLAG, que en resumen es un servicio de información bastante detallado. El servicio tiene muchos componentes que al trabajar en conjunto hacen que sea posible el funcionamiento del sistema. Existe un programa de uso exclusivo del cliente y otro para la administración de FLAG. Más adelante se irá explican a detalle los módulos de los cuales se conforma cada uno de éstos. Muchos son relativamente independientes unos de otros; no así la información que se genera ya que ésta se usa siempre en FLAG y constantemente está cambiando al ejecutarse el sistema.

Los módulos principales del programa del cliente son:

- la especificación de sus propias variables (las VCL);
- la inclusión de VMR en el modelo, que puede resultar en la solicitud de que el FLAG las ofrezca);
- la asignación de valores a sus VCL;
- la especificación de las fórmulas de su modelo;
- la formulación de los criterios de alarmas: selecciona variables críticas de su modelo, y especifica los rangos tales que si los nuevos valores cayeran fuera de ellos, desea ser advertido;
- especifica criterios de ejecución de su modelo: indica para las VMR que usa intervalos que actúan del mismo modo que los de alarma, excepto que ahora indican que desea que se recalculen las variables afectadas por ciertos cambios de valores en las VMR que indicó;

- y conjunto de consultas a su modelo, que le permitirán no sólo validar y completar su modelo, sino cumplen la función principal del FLAG: obtener la información resultante de las ejecuciones de su modelo.

Se incluye un modo de visualizar su modelo mediante una representación gráfica de las relaciones entre las variables, puesto que – especialmente en modelos con muchas variables – no es fácil determinar la corrección del modelo.

El programa central del FLAG – que se denomina el motor del FLAG puesto que de hecho realiza todas las acciones sustantivas de este servicio, consta de los siguientes programas:

- Obtiene valores nuevos de las VMR por medio de los agentes;
- Actualiza los valores de las mismas (que son los que usan los modelos);
- Ejecuta el modelo de FLAG (como se verá, se pueden incluir variables calculadas a partir de otras para ofrecer a los clientes);
- Aplica los criterios de ejecución aplicables a las variables cuyos valores cambiaron para todos los clientes;
- dispara la ejecución de los modelos indicadas por la aplicación de los criterios: esto a su vez resulta en la ejecución de todos esos modelos;
- evaluar los cambios en cada modelo de las variables que tienen alarmas y cuando proceda, generar avisos;
- enviar las alarmas generadas por el modo de comunicación aplicable.

El cliente obtiene la información actualizada vía sus variables, recalculadas con los cambios en las de su entorno de negocios. Puede consultar dichos valores constantemente o cuando le sea conveniente; en circunstancias excepcionales recibirá un aviso que le informa de algún cambio crítico, o una recomendación urgente de que consulte su modelo.

Para que esto ocurra FLAG debe ejecutar una serie de procesos indispensables para brindar un servicio confiable y eficiente. Los principales procesos que componen el FLAG, ilustrados en la Figura 8, son:

- *Variables del Mundo Real (VMR)*. En FLAG, al igual que en cada cliente, existe un modelo que se compone de variables (VMR), fórmulas y valores. Dichas variables son la información importante del sistema, por lo que deben de estar bien definidas para que el cliente pueda usarlas.
- *Comunicación*. Esta parte se agregó por el hecho de que ahora el sistema debe estar monitoreando la información de todos y cada uno de los clientes, debe tener disponible variables, valores, fórmulas y submodelos. La comunicación se encarga de mandar las bases de datos y archivos necesarios al sistema FLAG.

- *Agentes*. El módulo de agentes trabaja con las variables del mundo real; cada cierto tiempo, previamente especificado, busca los valores actuales de esas variables y los manda a FLAG.
- *Ejecución*. Esta es la parte más importante del sistema y del servicio, usa los procesos anteriores, debe leer los criterios de las VMR de cada cliente (que la comunicación provee) y evaluar si los valores traídos por los agentes provocan cambios significativos y por lo tanto disparar o no una ejecución de modelos. Al momento de la ejecución de un modelo de un cliente se van evaluando las alarmas y ahí es donde el cliente ve reflejado el funcionamiento del servicio.

Figura 8. Componentes principales del servicio FLAG

2.2. Terminología

Antes de entrar en el tema del sistema y su configuración, es imprescindible conocer el significado de algunos términos y explicar algunos aspectos para que los temas siguientes sean más comprensibles. En la Tabla 8 se describe la terminología.

Tabla 8. Terminología usada en FLAG

TÉRMINO	DESCRIPCIÓN
FLAG	Acrónimo de Flujo de Efectivo en Agronegocios
Modelo	Es un conjunto de fórmulas interrelacionadas por medio de variables y que forman una estructura de árbol.
Submodelo	Es un conjunto de variables de un modelo que cambian su valor siempre que la única de nivel cero haya cambiado.
Variable de cliente (VCLI)	En este tipo de variables el cliente define nombre, tipo y valores y puede asignarles una fórmula.
Variable del	Estas variables el cliente las agrega de las que están

mundo real (VMR)	disponibles en FLAG, el número original, nombre, tipo y valores están ya predefinidos, lo único diferente es el número con el que aparece en cada cliente que es variable.
Sinónimo	Es una variable virtual que se crea al momento de generarse los submodelos de un modelo. La creación de sinónimos es para que una variable no aparezca en más de una fórmula.
Agente de Búsqueda	Es un programa encargado de buscar la información en la red del vector de valores de cada variable del mundo real.
Cliente	Es una persona o una institución que contrata el servicio de FLAG, por lo que tiene acceso a las VMR y tiene un modelo con variables propias.
Variable escalar	Es una variable con un vector de 24 periodos de valores específicos escalares (i.e. no aleatorios).
Variable aleatoria	Es una variable con 24 periodos de valores aleatorios (con una distribución de probabilidad). En cada período hay un máximo de cuatro posibles valores de ese periodo con sus respectivas probabilidades.
Usuarios	Se refiere al personal físico que tendrá acceso al sistema. Entre éstos están los encargados de la administración de FLAG y los clientes.
Periodos	Es cada uno de los 24 valores de cada variable, sea escalar o aleatoria.
Periodicidad	Es el tiempo mínimo en el que un cliente desea que alguna de las VMR que está usando sea actualizada.

2.3. Lista de Módulos de FLAG

Los módulos que componen el sistema se dividen en dos tipos:

- Los que utiliza el cliente y
- Los exclusivos para la administración de FLAG.

Estos módulos se implementaron en programas separados, puesto que sus usuarios son distintos. De ese modo, está el FLAG para el cliente y el FLAG administrador.

El primero tiene todas las funciones con los que interactúa el cliente, mientras que el segundo contiene los programas que usan la información de cada cliente junto con la del mundo real.

2.3.1. Módulos del FLAG para el cliente

Se proporciona una lista de los submódulos de este módulo.

Altas, bajas y cambios de variables del cliente

Las variables son la unidad básica de los modelos: el cliente define las variables que usará su modelo. Cada variable tiene ciertos atributos, entre los cuales destacan el nombre de la variable, las unidades de medida de los valores y qué

tipo de periodos usará, pero hay otras propiedades que se detallarán más adelante.

Valores de las variables del cliente

El cliente proporciona los valores de sus variables “de abajo”, lo que significa que no tienen una fórmula. Para ello, hay funciones que le facilitan esta tarea. Cada variable tiene 24 valores correspondientes a 24 periodos.

Fórmulas

Las variables calculadas son aquéllas a las que se les asigna una fórmula. La fórmula puede contener variables con valor (de abajo), variables calculadas, constantes y variables del mundo real. Las fórmulas también tienen atributos que se utilizan en el proceso de ejecución, pero eso se detallará más adelante.

Altas, bajas y cambios de variables del mundo real que usa el cliente

Las VMR son las variables que FLAG pone a disposición del cliente para que las use en su modelo. En esta parte el cliente solo revisa las VMR que están disponibles y agrega las que crea necesarias. A éstas solo resta darles el valor de la periodicidad con la cual cada cliente necesite que sean actualizadas en su modelo ya que los demás datos como el nombre, unidades y tipo de periodos ya forma parte de la definición de la variable en el FLAG.

Criterios de alarmas

Los criterios para alarmas se definen exclusivamente sobre las variables calculadas del modelo del cliente; no se pueden indicar alarmas sobre variables aleatorias. Tampoco se pueden definir alarmas sobre una VMR, una constante o una variable que no tenga fórmula, puesto que ninguna de éstas cambiará de valor como resultado de una ejecución del modelo.

Los criterios se basan en el cambio que puede ocurrir cada vez que se ejecuta el modelo, es decir, si una variable cambia lo suficiente se genera una alarma. Este cambio puede ser en uno, varios o en los 24 periodos de la variable. El criterio se evaluará durante la ejecución de las fórmulas, y al final de la ejecución, si procede, se generará la alarma.

Criterios de ejecución

Sólo se pueden definir sobre variables del mundo real. La definición de los criterios es similar a las alarmas, excepto que es más sencilla. Observe que estos criterios le indican al motor del FLAG, que cuando las VMR cambian lo suficiente, se dispara una ejecución del modelo.

Ejecución de niveles y submodelos

Cuando el cliente terminó de elaborar (o modificar) las fórmulas de su modelo, mismas que residen en un archivo plano del sistema, se procede a “calcular” los niveles y submodelos. El nivel de una fórmula indica el orden en el cual se ejecutará: no se puede ejecutar una fórmula antes de calcular todas las de nivel inferior. Cabe agregar que este proceso modifica y amplía el archivo de fórmulas.

Con el fin de posibilitar una visualización del modelo, se impone una restricción que es transparente para el usuario (cliente u otro). Una variable no puede ser operando de más de una fórmula. Como esto en general no es el caso en el modelo que formula el cliente (puede tener varias fórmulas que tienen a alguna variable como operando) el FLAG usa un artificio que llama “sinónimos”. Cuando detecta que un operando aparece en una fórmula, pero ya estaba en otra, crea un sinónimo, es decir, otra variable que toma los mismos valores que la original. En la versión original del FLAG, esta actividad la realizaba el cliente mismo. Pero como causaba confusión y trabajo adicional, en la nueva versión del paquete se modificó este modo de incluir los sinónimos, y ahora los crea el programa de preparación de fórmulas, que se describe en esta sección.

En este módulo se genera el modelo tipo estructura de árbol de todas las fórmulas registradas y se le asigna un nivel a cada una de ellas (las que no son calculadas tendrán nivel 0) siendo la de mayor nivel una o más variables con fórmula que no pertenecen a otra fórmula. Como una VMR no puede tener una fórmula en el modelo del cliente, también tendrá nivel 0.

El concepto de *submodelo* es el siguiente: es la parte del modelo que se ve afectada por un cambio en una variable. En otras palabras, son las fórmulas que se tienen que ejecutar a consecuencia de un tal cambio. Para cada variable cuyos valores cambian, se tienen que ejecutar todas las fórmulas que la tienen como operando. Observe que habrá que buscar los sinónimos de esta variable como operandos, además de la fórmula que contenga la variable cuyos valores cambiaron.

El programa prepara, para cada VMR, el submodelo correspondiente. Lo hace creando una lista ordenada (por nivel) de las fórmulas que debe ejecutar cuando cambian los valores de esa VMR. Esto le permitirá reaccionar a un cambio de una variable del mundo real mediante la ejecución de su submodelo, en lugar de tener que recalcular todas las variables del modelo. En la sección donde se describen a detalle los submodelos también se explicará el concepto de “cambios aditivos” que contribuirán a la reducción de cálculos (y especialmente, de actividades de lectura de datos) en una ejecución del submodelo.

Este proceso se tiene que realizar cada vez que se modifique el modelo del cliente. De hecho, en el mismo paso se incorporan los criterios de alarma al archivo de fórmulas con el fin de evitar el uso de la base de datos (donde el cliente definió dichos criterios) a la hora de ejecutar el modelo y aplicar los criterios.

Ejecución del modelo

Un modelo de un cliente se ejecuta de dos formas posibles: por el mismo cliente o como consecuencia de cambios de valores de una o más VMR obtenidas por los agentes de búsqueda.

La ejecución de un modelo será descrita más abajo, como parte de las actividades del servicio FLAG: a la postre, es lo que le interesa a los clientes, que se valoren las variables calculadas de su modelo que reflejen cambios en los valores de las VMR que incluyó en el modelo.

Un modelo consta de fórmulas, las fórmulas de variables y éstas de valores. La ejecución de la preparación de las fórmulas, en especial el cálculo de niveles y el armado de los submodelos que se hace previamente crea un archivo en el que se almacena una lista de las variables con fórmula que se debe ejecutar y el orden en que se deben ejecutar. De ese modo, se ejecutan primero las fórmulas con variables de nivel 1, después las de nivel 2 y así sucesivamente hasta llegar al más alto nivel.

Consultas

El cliente tiene diversos modos de usar la información proporcionada por FLAG. Ya se comentaron las alarmas, que son avisos que le llegan acerca de circunstancias críticas. Pero también tiene la posibilidad de consultar “su modelo”: tanto la composición del mismo (qué variables incluyó, cuáles son las fórmulas) como, en especial, los valores de las variables.

El cliente puede hacer consultas cuando lo desee, por ejemplo para ver los valores de una u otra variable que sea de su interés. Adicionalmente, puede catalogar las consultas para no tener que volver a especificarlas en cada ocasión en que las use. Bastará invocar la consulta deseada y obtendrá los valores de las variables al día.

El cliente puede visualizar su modelo usando una facilidad del sistema. Le presenta el modelo de tal modo que puede ver los operandos de cada fórmula y de ese modo validar las que introdujo. Como la construcción de la gráfica requiere que se ejecute antes la preparación de fórmulas, se incluyó esta función en el módulo del administrador, a pesar de que el cliente dispone de dicha función.

2.3.2. Módulos del FLAG administrador

Dividiremos estos módulos en 2 grupos: los que se ocupan de OFRECER las diversas VMR y lo que denominamos el MOTOR del FLAG; que incluye las funciones con las que se obtienen los valores actualizados y ejecutan los modelos de los clientes.

2.3.2.1. La oferta de VMR del FLAG

La oferta de VMR por parte del FLAG consiste en definir variables que son de interés a sus clientes. Una variable puede ser incluida por solicitud de uno o varios clientes, o el FLAG mismo puede definir una fuente de información para que pueda ser aprovechada posteriormente por algunos de sus clientes.

Para ello, en forma análoga a lo que hace cada cliente, se definen variables por medio de sus diversos atributos. Se pueden agregar variables calculadas a partir de éstas (vía una fórmula) del mismo modo que en los modelos de los clientes. Naturalmente, estas fórmulas sólo podrán incluir como operandos otras VMR (y no de una variable de algún cliente). Los clientes pueden incluir estas VMR calculadas en sus respectivos modelos.

Finalmente, hay un módulo que permite especificar cómo se obtendrán los valores de las VMR (que no sean calculadas a partir de otras). Esto se hace indicando el AGENTE de BÚSQUEDA con el cual se consiguen valores actualizados a partir de alguna página o de otra fuente.

Altas, bajas y cambios de variables del mundo real

El módulo es similar al de ABC de variables del cliente. Hay un sinfín de candidatos posibles a variables que pueden integrarse al sistema, en un inicio pueden ser variables que se relacionen con el tipo de servicio que se está ofreciendo. En el ámbito agrícola las variables serían, por ejemplo, precio del jitomate, limón, aguacate, precio de insumos, precio de maquinaria agrícola, etc.

Especificación de los agentes de búsqueda

El sistema administrador de FLAG también tiene la función que permite definir agentes para la búsqueda de valores de las VMR existentes. Como especificaciones están las variables o variables a las que se encarga de actualizar, la ubicación (URL) del valor de la(s) misma(s) y la periodicidad mínima de búsqueda. Los agentes de búsqueda hacen que el servicio de vuelva dinámico y fidedigno ya que están en constante ejecución.

Tecleo de valores de las variables del cliente

La mayoría de las VMR obtendrán sus valores actualizados mediante el uso de agentes informáticos. Sin embargo, está la posibilidad de que haya algunas en las

que no sea posible o práctico asignarles valores por este medio, es decir, que los valores se obtengan de una fuente ajena a internet. En estos casos los valores se teclean manualmente como lo hace el cliente con sus variables.

Fórmulas entre variables del mundo real

Para poder ofrecer variables VMR calculadas a partir de otras, se pueden indicar fórmulas entre las VMR. Las fórmulas son similares a las que usan los clientes, con la salvedad de que no habrá sino VMR como operandos (y, una vez más, los sinónimos de las mismas).

Cálculo de niveles y submodelos

Siempre que haya cambios en el modelo, se tiene que ejecutar este módulo. Los cambios incluyen agregar y/o quitar operandos a fórmulas, agregar y/o quitar fórmulas. El modelo de tipo estructura árbol debe estar al día para que, al ejecutarse, los valores que ofrece el servicio siempre sean verosímiles. La explicación de estas funciones es la misma que para el modelo de un cliente, excepto que aquí no se agregan criterios de alarma puesto que no hay tales.

2.3.2.2. El motor del FLAG

Ejecución del modelo

Se hace desde el nivel inferior hasta el máximo de las variables con fórmula. Esta ejecución tendrá lugar siempre que haya VMR que funjan como operando de alguna fórmula.

Agentes

Los agentes informáticos son programas que se encargan de obtener los valores de las VMR. Antes de usar los agentes en las VMR, se deben de crear, es decir, darlos de alta mediante el programa de ABC de agentes. Al crear una VMR o después, es posible asignarle un agente que se va a encargar de obtener la información.

Comunicación

La comunicación es la parte del sistema que relaciona los modelos de los clientes con el FLAG administrador. La información de los modelos se almacena en bases de datos y archivos planos. Este módulo tiene como objetivo obtener la información de todos y cada uno de los clientes por lo que debe ser la versión más actual, es decir, el modelo más reciente modificado por ellos e igualmente con los valores de sus variables. Los criterios de alarmas y de ejecución también están incluidos.

Invocación de ejecuciones

Este módulo es el más importante de todo el sistema ya que es donde se conjunta toda la información y todos los procesos y de donde finalmente se obtiene el resultado del sistema: los avisos al cliente. Se encarga de evaluar los cambios de las VMR traídas por los agentes y determinar qué modelos de cliente deben ejecutarse.

Envío de alarmas

El módulo de alarmas consta de tres componentes principales: la captura de los criterios, la evaluación de los mismos (en la ejecución) y en envío de las alarmas. En esta última parte es cuando se manda un mensaje de la variable o variables y periodos que están siendo afectados en la actual ejecución. El mensaje se genera cuando se están evaluando los criterios y dependiendo de si cambia mucho es el tipo de aviso que se manda, es decir, si será un e-mail, un SMS o una llamada telefónica.

2.3.3. Proceso general del sistema

Entendiendo de forma general lo que hace cada módulo del FLAG, es posible ahora explicar el funcionamiento del sistema completo. Por medio de una serie de pasos se expondrá la forma y el camino que sigue el sistema para obtener los resultados que se esperan.

Actividades independientes a la ejecución del sistema

- *Captura de la información*. La información que el sistema necesita para que se ejecute el sistema es la siguiente:
    - Variables, valores, formulas, criterios de alarmas y criterios de ejecución, todo esto en la parte del cliente.
    - Variables del mundo real, valores y fórmulas en el modelo de FLAG.
    - Dar de alta los agentes que usarán las VMR existentes hasta el momento. Muchas variables se irán creando eventualmente así como los agentes que las alimentarán.

Es importante decir que el cliente puede hacer la captura de información en cualquier horario, lo mismo pasa con los usuarios del sistema administrador de FLAG. Lo que tiene que saber el cliente es que estos cambios no tendrán efecto ese día en su sistema sino hasta el siguiente día cuando FLAG se vuelve a actualizar.

- *Manejo de consultas*. La herramienta de consultas es un plus que hace que el sistema sea más atractivo porque es una manera fácil de enterarse del comportamiento de su modelo. Las consultas no tienen nada qué ver con el

funcionamiento del sistema ya que lo que hacen es acceder a las bases de datos y archivos de valores y mostrar la información.

- *Comunicación*. Antes de que comience una ejecución del sistema, éste debe asegurarse de que tiene a su disposición toda la información necesaria. Para esto se ejecuta un programa que verifica cambios en los modelos y criterios de los clientes, si hay cambios importa las bases y archivos y los guarda en el servidor de FLAG para que después los use en la ejecución. Este programa se ejecuta todos los días hábiles del sistema y siempre unos minutos antes de la ejecución principal, esto para que los datos que esté usando de los clientes sean siempre los más actuales.

Proceso de ejecución

La ejecución se efectúa diariamente, pero el operador del servicio puede cambiar esto si fuera conveniente. Inicia todos los días a primera hora y termina casi a la misma hora del día siguiente. Como se mencionó antes, este tiempo antes de la ejecución se usará para actualizar la información de los modelos de los clientes así como del mismo de FLAG. El proceso se hace de la siguiente manera:

- La ejecución inicia manualmente o de acuerdo a un procedimiento incluido en el sistema. En el primer caso, el usuario debe contar con el permiso para iniciar la ejecución.
- *Creación del archivo de criterios*. La información de los criterios de ejecución de todos los clientes está en una tabla de la base de FLAG denominada “criterios por cliente” la cual se genera cada vez que se importan datos de nuevos clientes, o bien, al actualizar información de los ya existentes. Para crear el archivo se debe leer dicha tabla, la información se almacena en estructuras específicas y después se guarda en el archivo. Es importante mencionar que este paso no siempre se ejecuta, solo se hace cuando el archivo aún no ha sido creado o cuando la información de criterios ha cambiado.
- *Lectura del archivo de criterios*. La información de los criterios de ejecución de todos los clientes se carga a memoria al inicio de la ejecución del programa. Para esto se generaron estructuras específicas manejadas en módulos de clase para que así fuesen leídas al momento de la evaluación de criterios.
- *Envío de agentes*. Éstos salen a buscar la información de valores de las variables. Las VMR tienen una periodicidad específica, un tiempo mínimo en el que los agentes deben actualizar sus valores. Cuando pasa este tiempo, el agente sale y vuelve a buscar el valor para esta variable.
- *Creación de archivo de agentes*. Cada vez que salen los agentes y vuelven con información, se crea una lista de las variables con valores pero solo los

de este ciclo. El archivo se guarda en el servidor de FLAG hasta que sea leído por otro programa.

- *Lectura del archivo de agentes*. Cuando el programa de lectura detecta que existe un archivo de agentes, lo lee, guarda la información en una lista en memoria y lo elimina.
- *Ejecutar el modelo FLAG*. Antes de invocar las ejecuciones de lo smodelos, hay que garantizar que si alguno de éstos ocupa una VMR calculada a partir de otras (VMR) los valores de la variable calculada reflejarán los cambios detectados por los agentes. Por lo tanto se ejecuta el modelo de las VMR; este hace que la lista de variables que hasta ahora está en memoria (indicando aquéllas cuyos valores cambiaron) puede crecer si se detectó durante la ejecución del modelo de FLAG alguna variable calculada cuyos valores cambiaron.
- *Evaluación de cambios*. Las VMR que están en la lista en memoria se evalúan una por una respecto a su valor anterior para cada modelo de los clientes. Esta evaluación de cada variable en cada cliente se hace por medio de los criterios de ejecución que previamente definió.
- *regWIN*. Para cada modelo que se tiene que ejecutar se graba un registro tipo regWIN (en la base de datos de Windows) en el que se graba el directorio correspondiente a ese cliente. Si la ejecución fue causada por cambios en una sola VMR de su modelo, se graba otro registro con el número de variable de la VMR en cuestión.
- *Se invoca la ejecución de modelos.* De ese modo, se le pasan los "parámetros" vía registros tipo regWIN.
- *Ejecución del modelo del cliente*. Cuando finalmente el proceso llega al modelo del cliente, éste se ejecuta y así el cambio en el mundo real se ve reflejado en el modelo del cliente.
- *Ejecución de alarmas*. Los criterios de alarmas que definió el cliente se hacen presentes en este punto del proceso. Los criterios se evalúan al momento de la ejecución de su modelo y al final las variables que tuvieron cambio significativo se les asigna un aviso que se le debe mandar al cliente.
- *Envío de los avisos*. En la base de datos de FLAG existe una tabla donde se guardan los avisos que en breve debe enviar a los clientes. Aquí está definido el tipo de aviso (e-mail, SMS o llamada), el cliente al cual se enviará y el contenido del mensaje.

Excepciones de la ejecución

Como en todo sistema, FLAG tiene ciertas excepciones que no está demás poner en claro y que tanto los usuarios como los clientes deben tomar en cuenta. Las siguientes son condiciones para las que la ejecución no se da de forma adecuada.

- *No hay datos*. La ejecución no se consuma si no hay datos de los clientes con los cuales trabajar. El cliente puede crear su modelo sin usar VMR, aunque es poco probable, es posible y en dichos casos el proceso de ejecución nunca afecta a este cliente. De hecho, nunca tendrá que ejecutar su modelo excepto cuando él mismo cambie algunos valores de sus variables de abajo.
- *El tamaño de los valores*. Los valores de todas las variables tienen un tamaño de memoria fijo y es de 32 bits, es decir, pueden guardar números de hasta 10 dígitos pero no mayores a $2^{31} – 1$ (aproximadamente 2 147 millones o menores a su equivalente negativo). Esto se definió así porque generalmente no hay cantidades de esos tamaños en la vida real, a menos que hubiese un modelo de la rama de astronomía, donde las cantidades utilizadas son, justamente, astronómicas. La realidad es que para FLAG lo común son variables de tipo *peso, altura, precios, toneladas*, etc. y para esto no es necesario usar muchos dígitos.

  Nota: las operaciones en este sistema se hacen con valores de tipo punto flotante. Los valores almacenados se convierten a *double* y se efectúa la operación. Los resultados son de mucho más dígitos que en el contemplado para el tipo *int32*, así que el valor se trunca hasta quedar con 4 decimales o 6 máximo si el valor no es muy grande, se aplica el factor y se almacena otra vez como número entero.

  Este manejo de los valores es una convención por parte de los diseñadores del sistema. Entre los motivos están el ahorro de espacio en memoria, agilización de la lectura y eficiencia del sistema.

2.4. Infraestructura

Los usuarios de este sistema se componen de varios tipos, el principal y que tiene acceso a todo es el administrador general. También están los que tienen uso limitado del sistema conforme al permiso que tengan. Entre estos últimos están los que se encargan de:

- ABC de nuevos clientes
- ABC de las variables del mundo real
- ABC de los agentes
- Actualizar los valores
- Generar el modelo
- Revisar los archivos de comunicación
- Monitorear los costos y precios de las VMR

Estos son los componentes más importantes que se deben tomar en cuenta en la infraestructura del sistema. Habrá usuarios para cada tipo de permisos de los mencionados anteriormente y otros de combinaciones de los mismos.

Traducción

Un aporte importante es la traducción del sistema ya que en su mayoría está implementado en el idioma español; sería muy conveniente tener la opción de usarlo en otros idiomas. Este módulo está contemplado actualmente pero aún no implementado, por lo que es una característica reservada a una versión futura del sistema.

3. CLIENTES

Un cliente de FLAG es una persona o empresa que decide utilizar los servicios de información ofrecidos, que incluyen de modo general:

- Ofrecer valores actualizados de ciertas variables del entorno de negocios del cliente (VMR)
- Ejecutar el modelo del cliente: esto consiste en recalcular variables en base a sus fórmulas, utilizando los valores actualizados de las variables de entorno
- Si procede, alertar al cliente de algún cambio significativo en indicadores definidos por el cliente (Alarmas).

De ese modo, hay funciones del sistema FLAG que usan los clientes, mientras que otras son de uso exclusivo del servicio, es decir, de personal que trabaja para ofrecer el servicio a sus clientes.

El cliente tiene las siguientes funciones:

- Crear su modelo
- Actualizar su modelo (indicar las variables y fórmulas que lo componen);
- Proporcionar los valores de “sus” variables (las variables del cliente que no se calculan con una fórmula, también llamadas variables “de abajo”);
- Indicar los criterios mediante los cuales FLAG sabe cuándo debe ejecutar el modelo como consecuencia de cambios en valores de las VMR que usa;
- Proporcionar criterios de alarma, lo que significa indicar algunas variables “críticas” y los rangos de variación de valores que considera que ameritan una advertencia inmediata;
- Consultar su modelo de numerosas maneras;
- Catalogar ciertas consultas que usará con frecuencia para no tener que definirlas en cada ocasión.

A continuación se describe cada uno de los procesos, incluyendo las interfaces ofrecidas, que permiten que un cliente realice las funciones de cada grupo.

3.1. Crear su modelo

Como el FLAG puede dar servicio a numerosos clientes, se establecen reglas de nomenclatura de directorios y archivos. Cada cliente usa una instancia de una base de datos típica llamada *MODELOBASE*, en la cual almacena su modelo y los criterios de ejecución y alarmas. Las consultas se almacenan en la tabla *QUERYCATALOGUE*.

Como se verá, además tiene tres archivos planos para almacenar información del modelo. En dos se almacenan los valores de las variables de su modelo, cuyos nombres son *Vectors.CLIVARS.DNVALUE* para variables aleatorias y *Vectors.CLIVARS.NUMVALUE* para las escalares. El último es *Formulae* donde se guarda la estructura de cada fórmula definida en el modelo con sus operaciones y operandos.

Los archivos correspondientes a un cliente residen en un directorio llamado “FLAG – Archivos” y las bases en otro de nombre “FLAG – Bases de datos”. A su vez, estos dos residen en “FLAG – Cliente” que está en la carpeta donde fue instalado inicialmente el sistema FLAG para ese cliente. Lo normal es que sea en “C:\” pero no es necesariamente el caso.

El servicio FLAG cuenta con la misma información anterior de cada cliente, es decir, tiene los archivos y bases de datos de cada cliente, en un directorio específico en el servidor FLAG llamado “CLIENTES”. Para cada cliente se crea una carpeta en dicho directorio, cuyos nombres consecutivos son CLIENTE0001, CLIENTE0002, CLIENTE0003, etc. hasta el número total de clientes que cuentan con el servicio. Es decir, a cada cliente se le asigna un número de cliente, que es el que se especifica en el nombre del directorio.

Los modos en los que un cliente puede trabajar en su modelo son dos: en su computadora o de forma remota. Si decide el primero, tendrá que “enviar” la base y los archivos de valores a FLAG cuando sufran modificaciones por algún cambio en el modelo, en los criterios o en los valores de las VCL. En el segundo caso no habrá problema ya que toda la información ya se encuentra alojada en el servidor de FLAG.

La base de datos del cliente está formada por varias tablas. Algunas de ellas se describen en las secciones dedicadas a su actualización.

3.2. Uso inicial del sistema por el cliente

Cuando un cliente comienza a usar el sistema FLAG, lo primero que debe hacer es generar una contraseña que será guardada en la base de datos en la tabla CLIENTE. Conforme genere su modelo y se creen los archivos de valores y fórmulas, la contraseña se modificará, esto es, generando palabras clave en dichos archivos y usando la contraseña inicial que había introducido. La nueva contraseña le será enviada por e-mail ya que no es común mantenerla en la base de datos.

Con la contraseña generada, se prosigue a introducirla en el campo correspondiente de la Figura 9.

Figura 9. Inicia el sistema FLAG

Se elige la opción que utilizará (por default está seleccionada “Modificar modelo”), se introduce el nombre y contraseña del cliente en los campos especificados en la Figura 9 y se presiona el botón “Acceder”. Lo anterior se diseñó pensando que el cliente puede usar una de las opciones del menú sin tener que entrar en las demás.

Modificar Modelo: Esta opción es la principal del sistema, es donde se crearán las fórmulas, criterios de alarmas, variables y se le dará valores a las mismas.

Ejecutar: el sistema se va directamente al módulo de ejecución.

Cambiar datos personales: esta parte es donde el cliente pone sus datos, modifica contraseña, entre otras cosas.

Consultar: se dirige al módulo de consultas

3.3. Actualizar su modelo

Primero se describen los tipos de variables que puede tener un modelo. Hay que recordar que las variables son vectores de 24 valores, que se denominan “períodos”. Las variables tienen dos “clasificaciones” independientes:

- Hay variables escalares y variables aleatorias.
- Hay variables de cliente (VCL) y variables del mundo real (VMR)

Las variables escalares tienen un valor para cada uno de sus 24 períodos (aunque un cliente puede decidir no usar los 24, siempre habrá 24 de ellos). En cambio las variables aleatorias tienen 24 variables aleatorias como periodos, cada una de las cuales tiene una distribución de probabilidad DISCRETA y de no más de 4 valores.

Cabe señalar que este número 4 no está basado en ningún criterio teórico o de otro tipo: es un parámetro de diseño del paquete FLAG, resultante de la decisión de que no se precisaban variables de más de 4 valores en el tipo de aplicaciones a las que se daría servicio. Este concepto se aclarará totalmente en la siguiente sección, dedicada a proporcionar los valores de las variables “de abajo” del modelo.

Las VMR son variables que ofrece el servicio para uso de los clientes que las necesiten (y por lo tanto, las usan en sus modelos). Éstas pueden ser de los dos tipos mencionados antes, es decir, escalares y aleatorias.

Este módulo se divide en tres elementos: la indicación de las VCL que usará el modelo, las VMR ofrecidas por el FLAG que el cliente decide incluir en su modelo; y la especificación de las fórmulas que permitirán el cálculo de algunas variables del modelo a partir de otras.

En las secciones dedicadas respectivamente a estos temas se detallarán las estructuras de datos que se utilizan para almacenar la información correspondiente.

3.3.1. Actualización de las VCL del modelo

El cliente describe (define) las variables “suyas” (a diferencia de las del mundo real) que usará en su modelo. Habrá variables de abajo (sin fórmula) y otras calculadas.

Antes de describir los procesos que usa un cliente para indicar sus variables, se describe la tabla de la base de datos llamadas “Variables del cliente” que se usa para almacenarlas. Las descripciones de los campos se aprovechan para proporcionar algunos detalles de lo que indica el cliente para cada VCL.

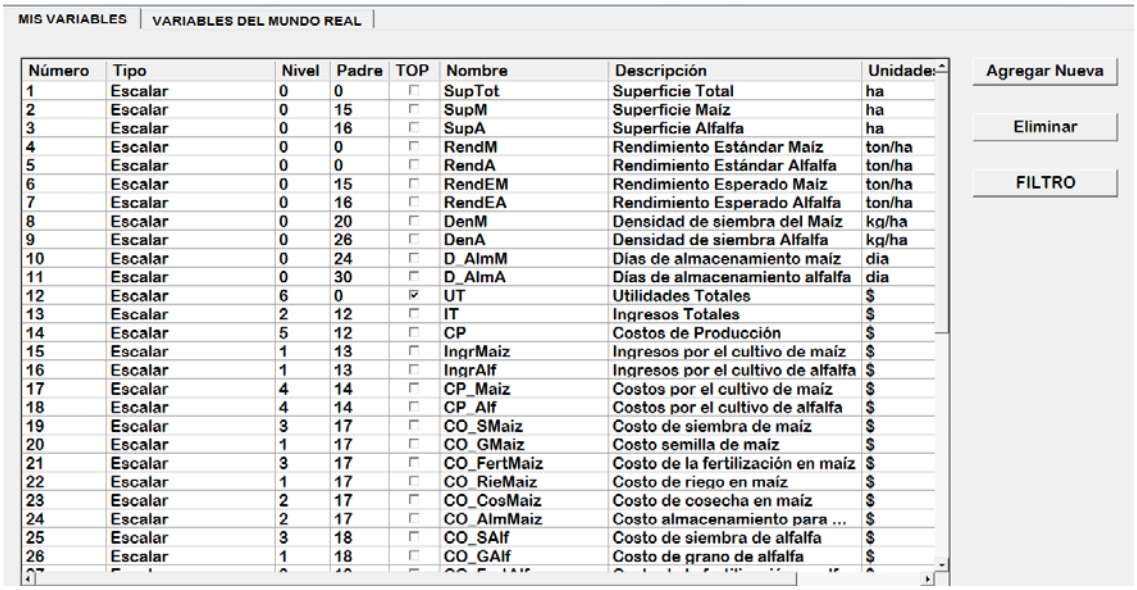
Tabla 9. Descripción de la tabla “variables del cliente”

CAMPO	TIPO	DESCRIPCIÓN
Número	Entero	ID consecutivo de la VCL que captura el cliente en su modelo. Entre 1 y 2000000
Tipo	Entero	1: variable; 2: constante
Tipo de valor	Entero	1: escalar; 2: aleatorio
Nivel	Byte	Nivel que ocupa en el modelo. Si es de abajo, el nivel es 0
Tiene fórmula	Booleano	Verdadero cuando es calculada
Submodelo	Texto	Número, en formato de texto, en el que se concatena el submodelo de la variable padre (dad) más su posición en dicha fórmula
Tamaño del submodelo	Byte	Número de caracteres del submodelo
Padre	Entero	Variable a cuya fórmula pertenece la actual variable
Es de arriba	Booleano	Verdadero si el dad de la variable es 0, es decir, que no pertenece a ninguna otra fórmula
Cuántos sinónimos	Byte	Cantidad de sinónimos de esta variable

Nombre	Texto	El cliente lo fija para identificar su variable
Descripción	Texto	Detalle que explica a la variable
Comentarios	Texto	Datos complementarios que el cliente quiera proporcionar
Unidades	Texto	Magnitud que especifica la variable
Factor aplicable	Entero	Número que identifica el número de decimales
Tipo de periodos	Entero	Horas, días, meses, años, …
Periodo inicial	Fecha	Fecha correspondiente al primer período

Para cada variable de cliente, los campos mostrados en la Tabla 9 deben contener información. La mayoría los define el cliente, sin embargo, otros son generados o actualizados por el sistema al calcular niveles y submodelos. Entre éstos destaca el submodelo, si es de arriba, la cantidad de sinónimos, el nivel y si tiene fórmula. Al dar de alta la variable, sus valores por default son cero (o “vacío” para los de tipo texto).

Cuando inicia la sesión aparecen dos pestañas del tabulador general de FLAG, una para VCL y la de VMR.



Número	Tipo	Nivel	Padre	TOP	Nombre	Descripción	Unidades
1	Escalar	0	0	☐	SupTot	Superficie Total	ha
2	Escalar	0	15	☐	SupM	Superficie Maíz	ha
3	Escalar	0	16	☐	SupA	Superficie Alfalfa	ha
4	Escalar	0	0	☐	RendM	Rendimiento Estándar Maíz	ton/ha
5	Escalar	0	0	☐	RendA	Rendimiento Estándar Alfalfa	ton/ha
6	Escalar	0	15	☐	RendEM	Rendimiento Esperado Maíz	ton/ha
7	Escalar	0	16	☐	RendEA	Rendimiento Esperado Alfalfa	ton/ha
8	Escalar	0	20	☐	DenM	Densidad de siembra del Maíz	kg/ha
9	Escalar	0	26	☐	DenA	Densidad de siembra Alfalfa	kg/ha
10	Escalar	0	24	☐	D_AlmM	Días de almacenamiento maíz	dia
11	Escalar	0	30	☐	D_AlmA	Días de almacenamiento alfalfa	dia
12	Escalar	6	0	☑	UT	Utilidades Totales	$
13	Escalar	2	12	☐	IT	Ingresos Totales	$
14	Escalar	5	12	☐	CP	Costos de Producción	$
15	Escalar	1	13	☐	IngrMaiz	Ingresos por el cultivo de maíz	$
16	Escalar	1	13	☐	IngrAlf	Ingresos por el cultivo de alfalfa	$
17	Escalar	4	14	☐	CP_Maiz	Costos por el cultivo de maíz	$
18	Escalar	4	14	☐	CP_Alf	Costos por el cultivo de alfalfa	$
19	Escalar	3	17	☐	CO_SMaiz	Costo de siembra de maíz	$
20	Escalar	1	17	☐	CO_GMaiz	Costo semilla de maíz	$
21	Escalar	3	17	☐	CO_FertMaiz	Costo de la fertilización en maíz	$
22	Escalar	1	17	☐	CO_RieMaiz	Costo de riego en maíz	$
23	Escalar	2	17	☐	CO_CosMaiz	Costo de cosecha en maíz	$
24	Escalar	2	17	☐	CO_AlmMaiz	Costo almacenamiento para ...	$
25	Escalar	3	18	☐	CO_SAlf	Costo de siembra de alfalfa	$
26	Escalar	1	18	☐	CO_GAlf	Costo de grano de alfalfa	$

Figura 10. Interfaz inicial del sistema FLAG para el cliente. Se muestran las VCL del modelo

La interfaz encargada de agregar y eliminar variables se puede ver en la Figura 10, la cual muestra la primera pestaña del tabulador general. Aquí aparece un *grid* (cuadro) que contendrá las variables que el cliente agregue; cada fila es una variable diferente, no importa que sea escalar o de distribución o una constante. Las variables del mundo real están en la otra pestaña, lo cual se explicará más adelante.

Cuando se quiera agregar una nueva variable, en la pestaña “Mis variables”, presiona al botón “Agregar nueva” y aparece la forma de la Figura 11. Aquí es

donde se elige si la nueva variable va a ser escalar o distribución y si será o no constante. Se “Acepta” y la nueva variable se agrega al final de la lista.

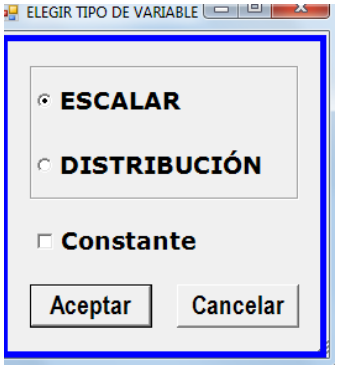


Figura 11. Tipo de variable a agregar

42	Escalar	1	30	☐	volAlf	Volumen de alfalfa cosechado	ton
43	Escalar	1	31	☐	diasAradM	Días que tarda la preparación...	dia
44	Escalar	1	37	☐	diasAradA	Días que tarda la preparación...	dia
45	Escalar	1	0	☑	[agregar texto]		[eleg]
1000001	Distribución Const	0	0	☑	[agregar texto]		[elegir]
1000002	Distribución	1	0	☑	Distr1	.	kg
1000003	Distribución	0	45	☑	Distr2	.	$
1000004	Distribución	0	100...	☑	Distr3	.	Hr

Figura 12. Nueva variable agregada

La variable agregada aparece en un cuadro como el que ilustra la Figura 12. Para indicar sus atributos, primero se selecciona de esta lista. En la Figura 13 se puede ver que al hacerlo se activan otras cuatro pestañas: una variable, valores, alarmas, fórmula. La que interesa ahora es la de “Una variable” porque es donde se le asignarán sus datos generales de la variable.

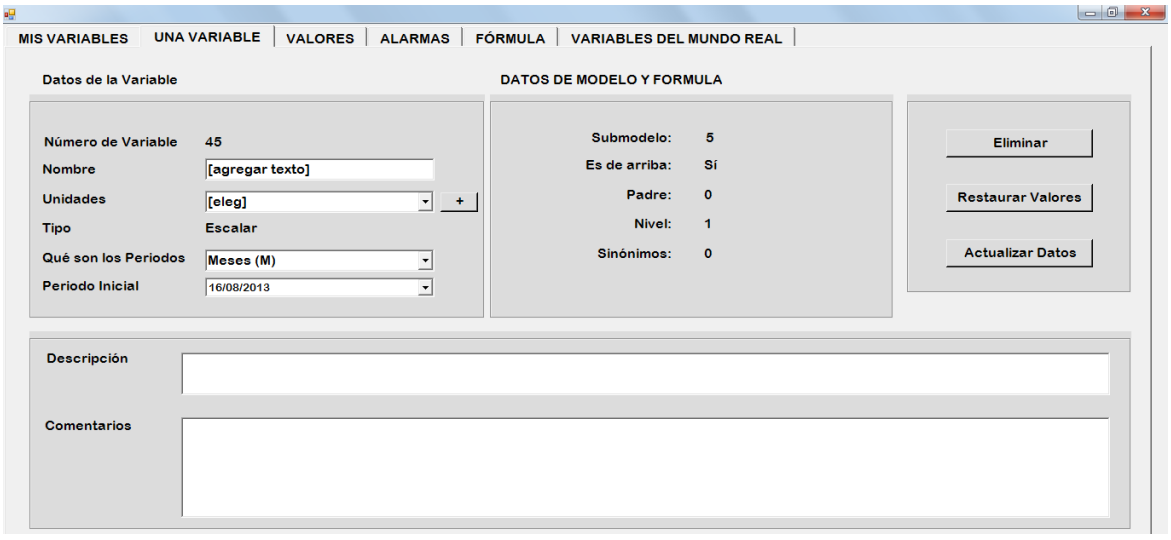
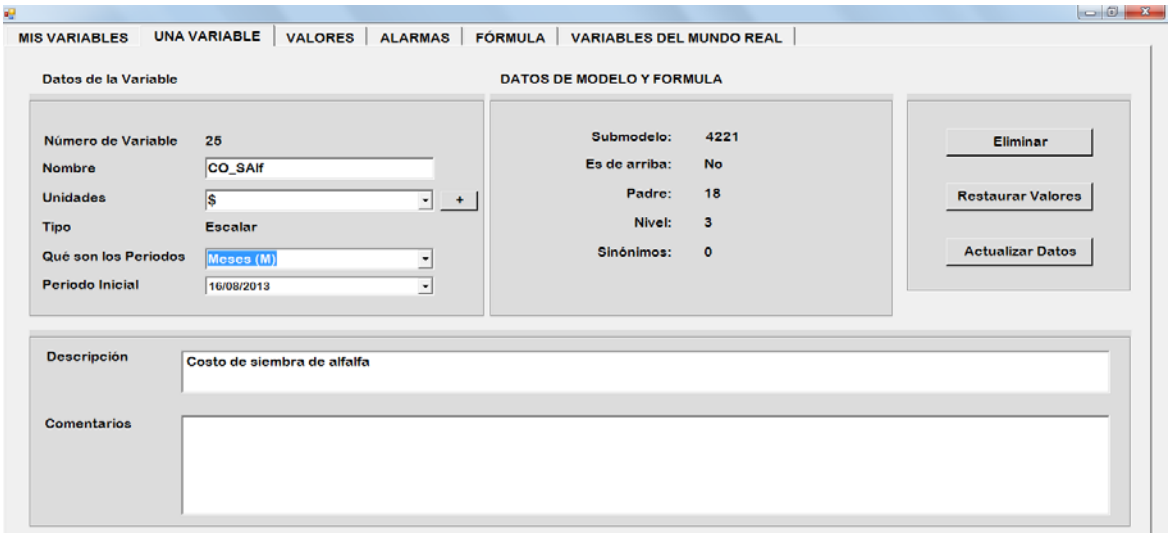


Figura 13. Información de una variable

Se llenan los campos con los datos preestablecidos por el cliente. Estos datos el cliente los determina ya que son para que él mismo identifique sus variables y le sirvan a la hora de asignar valores, hacer consultas, crear alarmas y crear fórmulas. En la Figura 14 se puede ver un ejemplo de una variable con información.

Figura 14. Una variable agregada



Eliminar variable

Para eliminar una variable no es suficiente seleccionarla y dar *click* en “Eliminar”. Antes se debe verificar si aparece en alguna fórmula (de hecho, conocer en cuáles aparece como operando). Si la usa alguna fórmula como operando no procede eliminarla. Un intento de hacerlo hará que el programa muestre las fórmulas en cuestión y le propone al usuario que elimine las variables en cuestión de dichas fórmulas. Observe que el sistema no realiza automáticamente estas actividades porque el cliente perdería control sobre su modelo.

3.3.2. Actualización de las VMR del modelo

FLAG ofrece ciertas VMR a sus clientes. La selección de esas variables en general se deberá a que algún cliente las solicitó; naturalmente otros clientes también podrán utilizarlas. Hay un módulo del FLAG que permite introducir las VMR que se ofrecen, pero un cliente no puede hacerlo: es una función reservada a personal del servicio FLAG. De ese modo, el cliente selecciona las VMR que le interesan o necesita de una lista de todas las VMR que ofrece el FLAG.

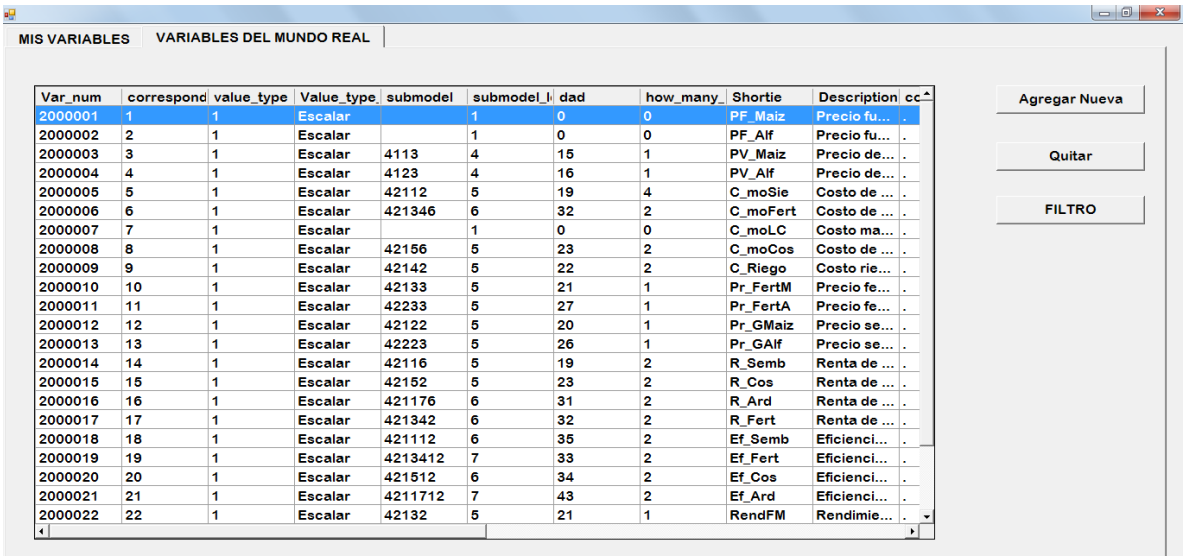
Se describe la tabla de la base de datos en las que se guardan las VMR elegidas, y una vez más se aprovecha la descripción de los campos de la tabla para detallar algunos conceptos de las mismas.

Tabla 10. Descripción de la tabla “variables del mundo real del cliente”

CAMPO	TIPO	DESCRIPCIÓN
Número	Entero	ID consecutivo de la VMR que el cliente agrega a su modelo. Entre 2,000,001 y 4,000,000
Corresponde a		Número de la VMR en FLAG
Tipo de valor	Entero	1: escalar; 2: aleatorio
Tiene fórmula	Booleano	Verdadero cuando es calculada
Submodelo	Texto	Número, en formato de texto, en el que se concatena el submodelo de la variable padre (dad) más su posición en dicha fórmula
Tamaño del submodelo	Byte	Número de caracteres del submodelo
Padre	Entero	Variable a cuya fórmula pertenece la actual variable
Es de arriba	Booleano	Verdadero si el dad de la variable es 0, es decir, que no pertenece a ninguna otra fórmula. Esto indica que la VMR no se usa en el modelo del cliente excepto como información complementaria.
Cuántos sinónimos	Byte	Cantidad de sinónimos de esta variable
Nombre	Texto	Es el nombre proporcionado por FLAG a esta VMR
Descripción	Texto	Detalle que explica a la variable
Comentarios	Texto	Datos complementarios que el cliente quiera proporcionar
Unidades	Texto	Magnitud que especifica la variable
Factor aplicable	Entero	Número que identifica el número de decimales
Tipo de periodos	Entero	Horas, días, meses, años, …
Periodo inicial	Fecha	Fecha correspondiente al primer período
Unidades de periodicidad		Es la magnitud que FLAG tomará en cuenta para que esta VMR sea actualizada para el cliente actual. Pueden ser minutos, horas, días
Mínima periodicidad		Es el tiempo mínimo para que esta VMR sea actualizada por FLAG.

Para el caso de las VMR, la mayoría de la información ya viene determinada por FLAG desde el momento en que se agrega al modelo (ver Tabla 10). Al igual que con las VCL, el submodelo se determina al ejecutarse el programa que los calcula. El dato importante que el cliente debe especificar para cada VMR que desea utilizares la periodicidad.. Dicha información es importante ya que con esto se ejecutarán los criterios de forma correcta y en el tiempo indicado.

En este sistema cuando no se ha seleccionado ninguna variable de cliente la segunda pestaña corresponde a la parte de variables del mundo real que usa en su modelo.

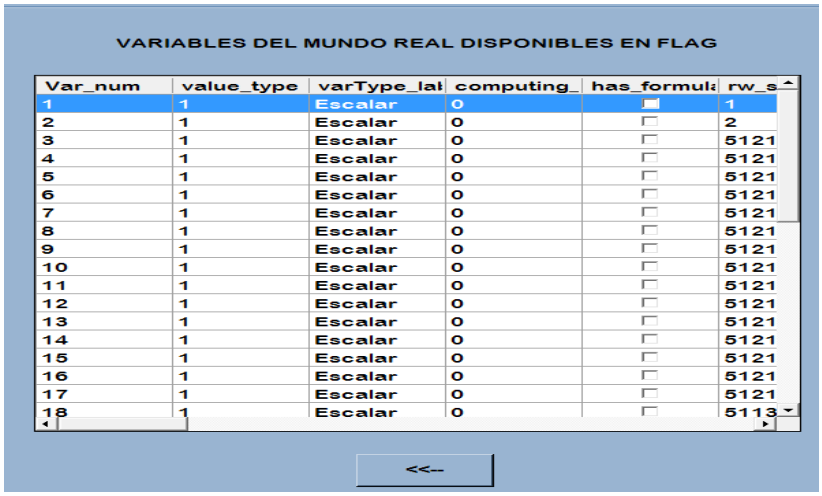


Var_num	correspond	value_type	Value_type_	submodel	submodel_l	dad	how_many_	Shortie	Description	cc
2000001	1	1	Escalar		1	0	0	PF_Maiz	Precio fu...	.
2000002	2	1	Escalar		1	0	0	PF_Alf	Precio fu...	.
2000003	3	1	Escalar	4113	4	15	1	PV_Maiz	Precio de...	.
2000004	4	1	Escalar	4123	4	16	1	PV_Alf	Precio de...	.
2000005	5	1	Escalar	42112	5	19	4	C_moSie	Costo de ...	.
2000006	6	1	Escalar	421346	6	32	2	C_moFert	Costo de ...	.
2000007	7	1	Escalar		1	0	0	C_moLC	Costo ma...	.
2000008	8	1	Escalar	42156	5	23	2	C_moCos	Costo de ...	.
2000009	9	1	Escalar	42142	5	22	2	C_Riego	Costo rie...	.
2000010	10	1	Escalar	42133	5	21	1	Pr_FertM	Precio fe...	.
2000011	11	1	Escalar	42233	5	27	1	Pr_FertA	Precio fe...	.
2000012	12	1	Escalar	42122	5	20	1	Pr_GMaiz	Precio se...	.
2000013	13	1	Escalar	42223	5	26	1	Pr_GAlf	Precio se...	.
2000014	14	1	Escalar	42116	5	19	2	R_Semb	Renta de ...	.
2000015	15	1	Escalar	42152	5	23	2	R_Cos	Renta de ...	.
2000016	16	1	Escalar	421176	6	31	2	R_Ard	Renta de ...	.
2000017	17	1	Escalar	421342	6	32	2	R_Fert	Renta de ...	.
2000018	18	1	Escalar	421112	6	35	2	Ef_Semb	Eficienci...	.
2000019	19	1	Escalar	4213412	7	33	2	Ef_Fert	Eficienci...	.
2000020	20	1	Escalar	421512	6	34	2	Ef_Cos	Eficienci...	.
2000021	21	1	Escalar	4211712	7	43	2	Ef_Ard	Eficienci...	.
2000022	22	1	Escalar	42132	5	21	1	RendFM	Rendimie...	.

Figura 15. Interfaz de variables del mundo real

En la Figura 15 se pueden ver las variables que el cliente tomó del FLAG administrador para usarlas en su modelo. El campo “corresponde a” viene directo del sistema administrador, es el número original que tiene, pero cuando se importa dicha variable al modelo del cliente se le asigna otro número automático que, como se indicó antes, es mayor a 2000000 y menor a 3000000 si es escalar y si es mayor a 3000000 es variable aleatoria. El cálculo de los campos “submodelo”, “tamaño del submodelo”, “cuantos sinónimos” y otros, se explicarán en el apartado de cálculo de niveles y submodelos.

Para dar de alta nuevas variables del mundo real que ya existen en FLAG administrador, hay que dar click en el botón “Agregar Nueva” de la Figura 15 y aparecerá un cuadro con el de la Figura 16.



Var_num	value_type	varType_lal	computing_	has_formul	rw_s
1	1	Escalar	0	☐	1
2	1	Escalar	0	☐	2
3	1	Escalar	0	☐	5121
4	1	Escalar	0	☐	5121
5	1	Escalar	0	☐	5121
6	1	Escalar	0	☐	5121
7	1	Escalar	0	☐	5121
8	1	Escalar	0	☐	5121
9	1	Escalar	0	☐	5121
10	1	Escalar	0	☐	5121
11	1	Escalar	0	☐	5121
12	1	Escalar	0	☐	5121
13	1	Escalar	0	☐	5121
14	1	Escalar	0	☐	5121
15	1	Escalar	0	☐	5121
16	1	Escalar	0	☐	5121
17	1	Escalar	0	☐	5121
18	1	Escalar	0	☐	5113

Figura 16. Variables del mundo real en FLAG

Aquí elegimos la variable y al dar *click* en botón "<<--" inmediatamente se agregará a la base de datos del cliente con el número consecutivo que le corresponde. Ahora se puede ver la nueva variable en nuestra lista del sistema.

Información de una VMR

Todas y cada una de las variables que se anexaron al modelo de un cliente tienen la información que se muestra en la Figura 17.

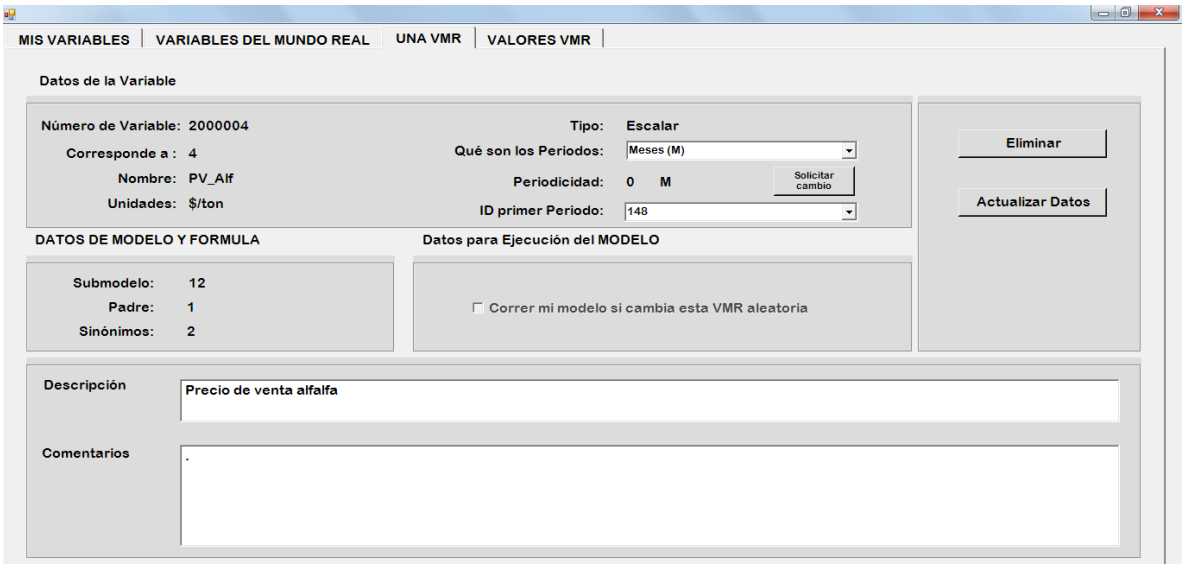


Figura 17. Información de una VMR del cliente

La mayoría de estos datos son fijos y no se pueden cambiar, por ejemplo el número es el que se le asigna al importarse al modelo, debe ser mayor a 2 millones. El valor de "corresponde a" es el número que tiene la variable en FLAG. El nombre, unidades y tipo también vienen de FLAG.

Hay un proceso que se debe seguir para cambiar la periodicidad de cualquier VMR que esté en el modelo. Cuando se da de alta una VMR en el modelo, ésta tiene como periodicidad un día, es decir, se conseguirán valores de esa VMR cada día. Este valor se proporciona por *default*; si se quiere modificar se debe hacer de forma manual en el sistema, eligiendo una de las periodicidades disponibles y después esperar a que el servicio lo tome en cuenta al final de día y al siguiente se le avisará de la situación. Se le enviará un e-mail con detalles sobre la solicitud para que esté al tanto del proceso. En general un periodo menor de actualización resultará en un costo mayor, pero este componente (el llamado *pricing*) no se ha desarrollado en esta versión del FLAG.

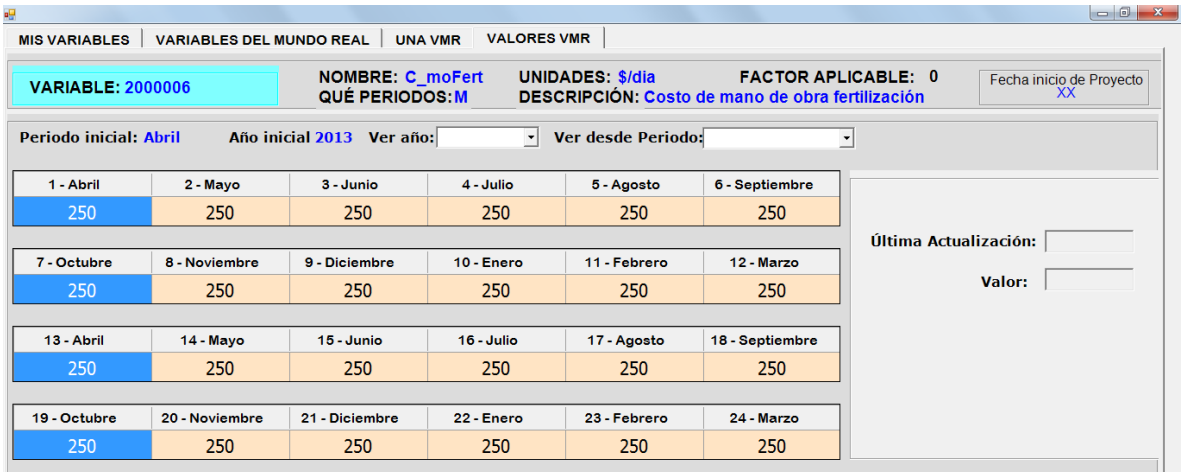


Figura 18. Valores de una VMR

Los valores de las VMR son siempre fijos (ver Figura 18), en el tabulador aparecen solo para información del cliente siempre que la selecciona. La interfaz es semejante a la de valores de variables del cliente. En la parte de arriba están los datos generales de la variable, después los valores de los 24 periodos y en la parte de la derecha la fecha de la última actualización y el valor que llegó. Observe que la facilidad de indicar otros valores sólo se incluyó para efectos de simulación tipo “what if”: el cliente puede determinar el impacto sobre sus variables de un cambio simulado de valores de alguna o todas las VMR que utiliza en su modelo.

Para eliminar una de estas variables ocurre lo mismo que en el caso de las de cliente, es decir, se buscan las variables cuyas fórmulas contengan dicha variable como operando y se informa al cliente eso para que la quite de la o las fórmulas para que pueda eliminar la variable.

3.3.3. Actualización de las fórmulas del modelo

Como se ha explicado, la naturaleza del servicio FLAG consiste esencialmente de ejecutar los modelos de sus clientes con valores actualizados de ciertas variables. De ese modo, se puede decir que uno de los elementos más importantes del FLAG es el modelo de cada uno de sus clientes.

Una fórmula indica una serie de operaciones (matemáticas o estadísticas) que se efectúan entre otras variables del modelo para obtener los valores de la variable a la cual se asocia la fórmula. Es decir, las variables calculadas del modelo tendrán una fórmula. Quizá no sea redundante señalar que una VMR del modelo no puede tener una fórmula, puesto que sus valores los proporciona el servicio de actualización de las VMR.

Las fórmulas se guardan en un archivo plano. Se prefirió esta estructura a la del uso de una tabla de la base de datos por diversos motivos entre los cuales está:

- Menos espacio en disco. Esto no es significativo excepto cuando se trata de transmitir las fórmulas entre computadores, especialmente por Internet;
- Ofrece numerosas ventajas en cuanto a programación cuando se usa el archivo, especialmente en el programa que prepara las fórmulas para su ejecución (que se describe en un capítulo por separado).
- Las tablas para almacenar las fórmulas serían de tres niveles. La fórmula misma (que pudiera formar parte de la tabla de las variables, pero eso haría que aún las variables de abajo tendrían esos campos sin utilizarlos). Luego están los subtotales; para cada subtotal, otra tabla tendría las operaciones y operandos. Esta situación no es grave, pero no ofrece ventaja alguna sobre el uso de un archivo plano en el que se guarda la fórmula con una estructura que incluye todos sus campos.

En esta sección, se describen primero las fórmulas (en qué consisten, cómo se definen), el modo de introducirlas al sistema y, finalmente, se documenta la estructura del archivo que las contiene. Se adoptó este orden porque de lo contrario muchos de los campos del archivo no serían interpretables antes de las explicaciones de las fórmulas.

3.3.3.1. Cómo se define una fórmula en FLAG

Uno de los problemas a los que se enfrenta un usuario que define una fórmula es la notación y el hecho de que las operaciones se deben efectuar en el orden correcto. Para ello, se pueden usar (esencialmente) 3 métodos:

- Se indica con diversos niveles de paréntesis, corchetes o similares las agrupaciones de operaciones que se deben realizar antes de ejecutar otras;
- Se usa la notación posfija, anteriormente llamada notación polaca y que se hizo popular desde los años 50 (Arthur et al, 1954);
- Algún modo *ad hoc* que se define, en especial tomando en cuenta el tipo de usuarios potenciales de las funciones que se usan para definir una fórmula.

El uso de paréntesis puede ser confuso para ciertos usuarios, en especial cuando hay varios niveles de anidamiento. Se pueden diseñar interfaces que contemplen esta dificultad, pero de todos modos son incómodas. Bauer (el que diseñó el FLAG) ha tenido malas experiencias en este tipo de modelos definidos por usuarios.

La notación posfija estuvo en boga durante mucho tiempo. Su uso implicaba que el usuario tenía que conocerla, o invertir un esfuerzo considerable en aprenderla y entender: aún después de hacerlo, las fórmulas resultantes tenían muchos errores en muchas aplicaciones.

Por lo tanto, se decidió usar un modo basado en subtotales, que reemplazan a los paréntesis. Se agrupan operaciones que se deben hacer en forma conjunta en un subtotal, y los siguientes subtotales los usan como argumentos (u operandos, como se denominan en FLAG). A pesar de que se describen las fórmulas en forma "teórica", a muchos lectores les será más fácil entenderlas con los ejemplos proporcionados más abajo.

De ese modo, una fórmula consiste de (hasta) 4 SUBTOTALES. El último subtotal utilizado es el resultado, es decir, la variable calculada tomará los valores proporcionados por dicho subtotal.

Cada subtotal tiene un máximo de 4 OPERANDOS (que pueden ser Variables o uno de los subtotales "anteriores"). Los operandos están unidos por OPERACIONES (se verá que hay excepciones a esta regla, aunque sólo se debe a una notación y no a una diferencia conceptual).

Las operaciones implementadas son: suma, resta, producto, división, máximo y mínimo. Estos dos últimas no aplican a operandos aleatorios. Además, se incluyen las operaciones estadísticas Media, Moda, Mediana, Varianza y Desviación estándar para argumentos aleatorios.

Proporcionamos algunos ejemplos de fórmulas, con la observación de que las variables (indicadas por el número de variable) no tienen ningún significado. Indicamos primero "lo que queremos que haga la fórmula" y luego cómo se introduce al FLAG. Agregamos que no se proporcionan explicaciones "matemáticas" especialmente relacionadas con el orden en el que se deben ejecutar las operaciones debido a la naturaleza de los lectores potenciales de esta tesis.

EJEMPLO 1: VCL023 = VCL003 + VMR040

Cuantos subtotales usa: 1

Subtotal (1) = VCL003 + VMR040

EJEMPLO 2: VCL028 = VCL080 + VMR040 * VCL009

Cuantos subtotales usa: 2

Subtotal (1) = VMR040 * VCL009

Subtotal (2) = VCL028 + Subtotal (1)

EJEMPLO 2: VCL028 = Mínimo (VCL080, (VMR040 * VCL009))

Cuantos subtotales usa: 2

Subtotal (1) = VMR040 * VCL009

Subtotal (2) = Mínimo (VCL080, Subtotal (1))

Observe que ninguna de las variables que aparecen como operandos de la operación mínimo (o máximo) puede ser aleatoria: no tiene sentido (o habría que definir este tipo de operaciones entre una variable aleatoria y otra que, excepto en algunos casos, no resultan de interés).

Una fórmula puede contener un máximo de 13 variables por el hecho de que hay solamente 4 subtotales con cuatro operandos posibles en cada uno, pueden capturarse las 16 variables sin embargo, como ya se dijo, el resultado de la fórmula es el último subtotal y si en él hay 4 variables, los otros subtotales no tendrán impacto en el resultado.

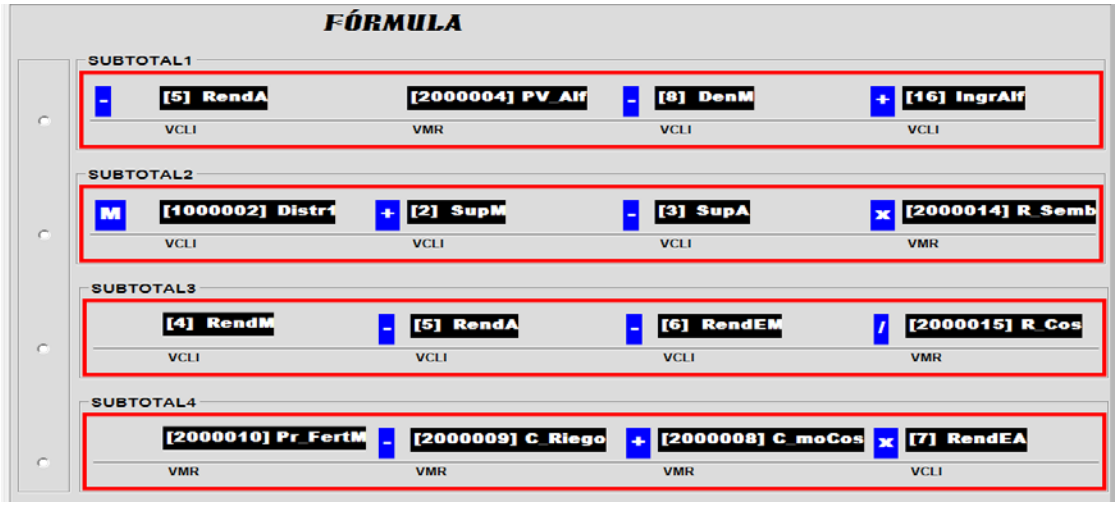


Figura 19. Ejemplo de fórmula mal capturada

FLAG tiene un sistema que evalúa la captura de fórmulas. Por ejemplo, cuando el cliente crea una fórmula semejante a la de la Figura 19, el programa no permite guardarla hasta que sea modificada y tenga más sentido. En el caso de la figura, hasta que haga uso correcto de los subtotales le será permitido almacenar esta fórmula. En la siguiente se puede ver una fórmula con el máximo de variables usadas correctamente. En especial, no se admiten fórmulas que contienen subtotales que no aparecen en uno posterior (excepto el último subtotal indicado).

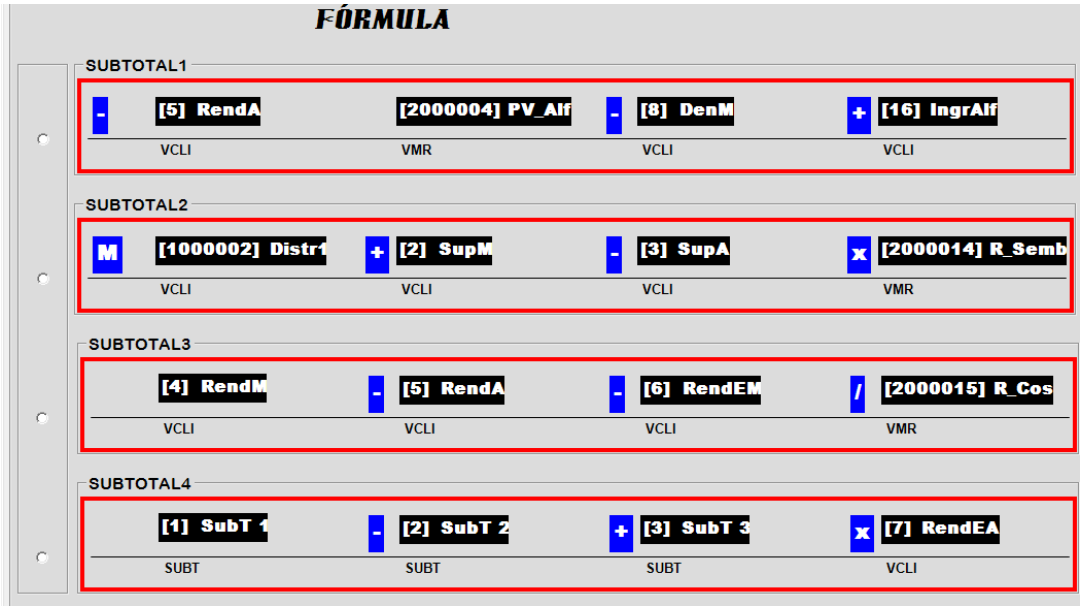


Figura 20. Ejemplo 1 de fórmula capturada correctamente

En la Figura 20 los operandos 1, 2 y 3 del subtotal 4 son los primeros tres subtotales, lo que significa que las variables de éstos sí son tomados en cuenta; también se puede ver que en total son 13 variables posibles, al igual que en el siguiente ejemplo:

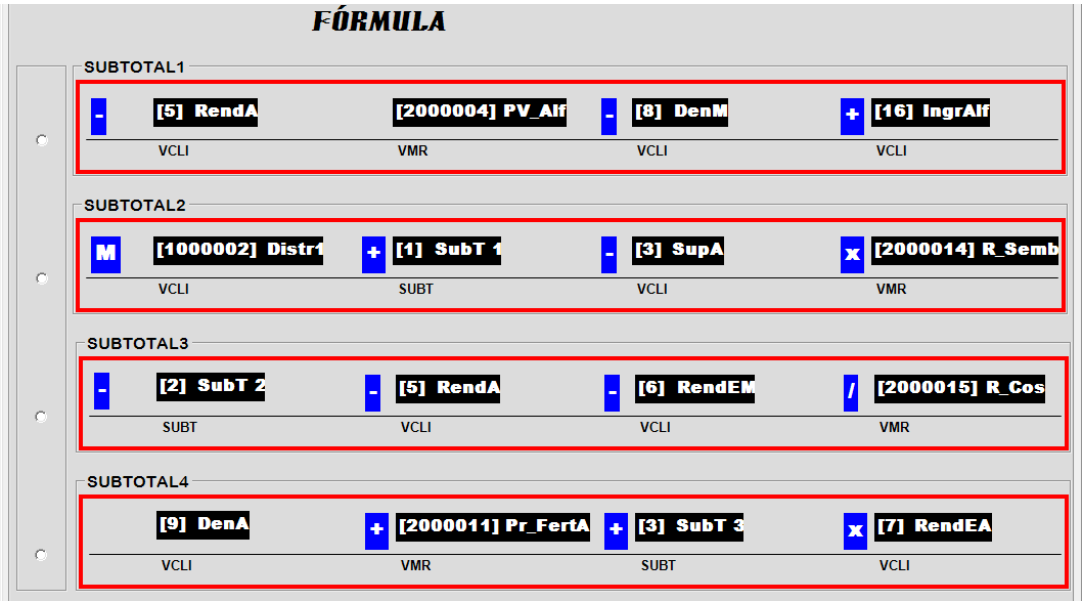


Figura 21. Ejemplo 2 de fórmula capturada correctamente

En la Figura 21 se puede ver que en el subtotal 2 aparece el 1 como operando, en el subtotal 3 aparece el 2 y en el 4 está el subtotal 3 como operando, por lo que todas las variables son tomadas en cuenta. Se puede ver que hay en total 13 variables.

Una fórmula siempre corresponde a una variable y ésta a su vez estrictamente es una VCLI y ésta tiene que ser escalar o aleatoria, es decir, como se haya definido a la hora que se crea.

Algo muy importante que se debe tomar en cuenta a la hora de capturar las fórmulas es el saber de qué tipo es la variable en cuestión y tener cuidado de los tipos que se estén usando como operandos. Por ejemplo, una variable originalmente creada como escalar que contenga sólo operaciones entre variables aleatorias es contradictoria porque el resultado sería una variable aleatoria también. Observe que las operaciones estadísticas sólo pueden tener un operando (que forzosamente debe ser aleatorio).

El proceso que evalúa que la captura se haga de manera correcta no solo toma en cuenta el uso de subtotales sino también el orden de las operaciones – dentro de un subtotal – y verifica el tipo de resultado que se obtiene en cada subtotal, es decir, si es escalar o aleatorio.

Subtotal

Un subtotal tiene la misma función que tienen los paréntesis en las operaciones matemáticas, es decir, solamente el de agrupar operaciones. Se usa para operaciones de multiplicación y división ya que éstas se deben ejecutar antes que la suma y resta. También se usa cuando la fórmula tiene muchas variables y es necesario dividir la fórmula y ejecutarla por partes. Usar subtotales será indispensable para los casos antes mencionados.

Cada subtotal tiene como máximo cuatro operandos los cuales pueden ser variables de cliente, del mundo real, constantes u otro subtotal anteriormente creado.

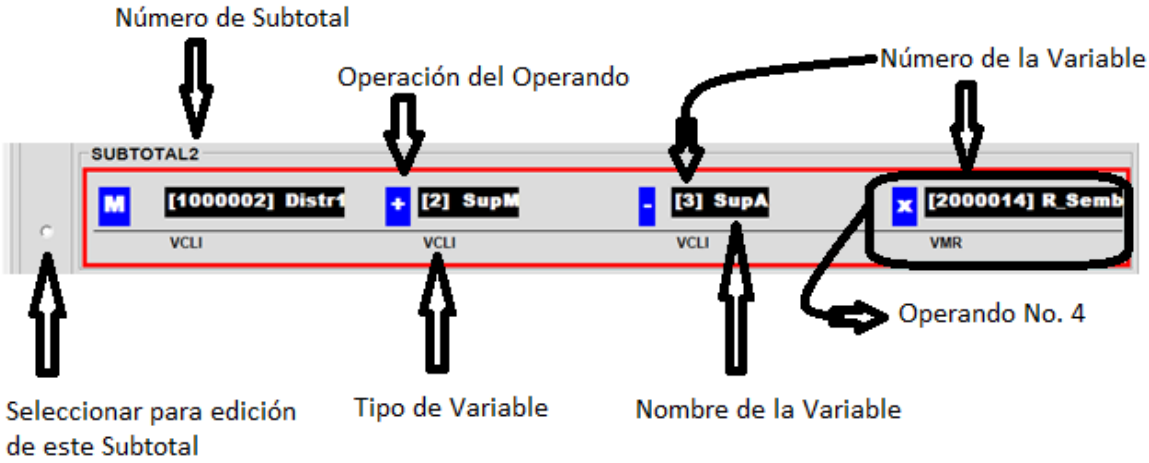


Figura 22. Estructura de un subtotal

La cuestión del subtotal en la Figura 22 es que si en una fórmula hay operaciones de producto y división, éstas tienen que calcularse primero en un subtotal aparte y después agregarlas a las sumas y restas que existan.

Operando

Las partes que componen un operando son las siguientes:

Operación. Es la operación matemática o estadística que afectará a los valores del operando en cuestión. Son un total de 13 operaciones disponibles que se pueden elegir dependiendo del tipo de variable del operando y de su posición en el subtotal; para esto existen ciertas reglas. Las operaciones son: suma, resta, multiplicación, división, raíz cuadrada, cuadrado, máximo, mínimo, media, mediana, moda, desviación estándar y varianza estimada.

Tabla 11. Operaciones a elegir para los operandos

Posición del operando en el subtotal	Tipo de variable en el operando	Operaciones admitidas
Posición Inicial	*Escalar*	Resta, máximo, mínimo, raíz cuadrada y cuadrado
	Aleatoria	Resta, media, mediana, moda, desviación estándar y varianza estimada.
Posición NO Inicial	*Escalar y Aleatoria*	Suma, resta, multiplicación y división

La opción “vacío” está disponible también para los operandos iniciales, prácticamente esta será la opción más común al inicio de un subtotal. Con las restricciones mostradas en la Tabla 11 será más difícil cometer errores por el cliente cuando cree sus fórmulas. De esta forma un operando en la posición número 2 del subtotal no puede tener como operación una de tipo estadístico, máximo, mínimo, cuadrado o raíz. Esto se podrá corroborar a la hora de la captura.

Tipo de Variable. Puede ser uno de los siguientes:

- *VCLI*: variable de cliente
- *VMR*: variable del mundo real del cliente
- *CNTE*: constante, se crea igual que una VCLI pero carece de ciertas propiedades
- *SUBT*: subtotal previamente generado. Si se está modificando el 1 no tiene opciones a elegir, si es el 2 aparece el 1 como opción, si es el 3 se puede elegir el 1 o el 2 y si es el subtotal 4 el que se está modificando, las opciones son los tres primeros.

Operando. Es el dato del cual se tomarán los valores para aplicarle la operación que se indicó. Puede ser de cualquiera de los cuatro tipos de variable mencionados antes.

Desfase. Cuando se ejecuta una fórmula los valores se ejecutan homólogamente con el periodo que le corresponde entre una variable de un operando y la siguiente, es decir, el periodo 1 con el 1, el 2 con el 2 y así sucesivamente. Lo que hace esta opción es que si por ejemplo se elige un desfase de +1, de la variable anterior o el acumulado, se aplicará el periodo 1 con el 2 de la actual, el 2 del anterior con el 3 del actual, el 3 de anterior con el 4 del actual y así sucesivamente; si el desfase es de -5 será entonces el periodo 1 del anterior con el 20 del actual, el 2 del anterior con el 21 del actual y así sucesivamente.

El operando es la entidad básica de la fórmula a diferencia del subtotal que solamente simula un paréntesis en un conjunto de operaciones. Las variables que conforman una fórmula están en los operandos, al cual se le aplica una operación y un desfase.

3.3.3.2. Captura

Para capturar fórmulas primero se activa la pestaña de “fórmulas” al seleccionar cualquier variable.

Figura 23. Interfaz principal de captura de fórmulas

Como se puede ver en la Figura 23, no hay fórmula capturada en un inicio, es decir, el espacio está vacío. Para iniciar hay que agregar subtotales con el botón “+” que está en la parte de la derecha, también se pueden quitar con el botón “-”. En la Figura 24 se ven los 4 subtotales agregados.

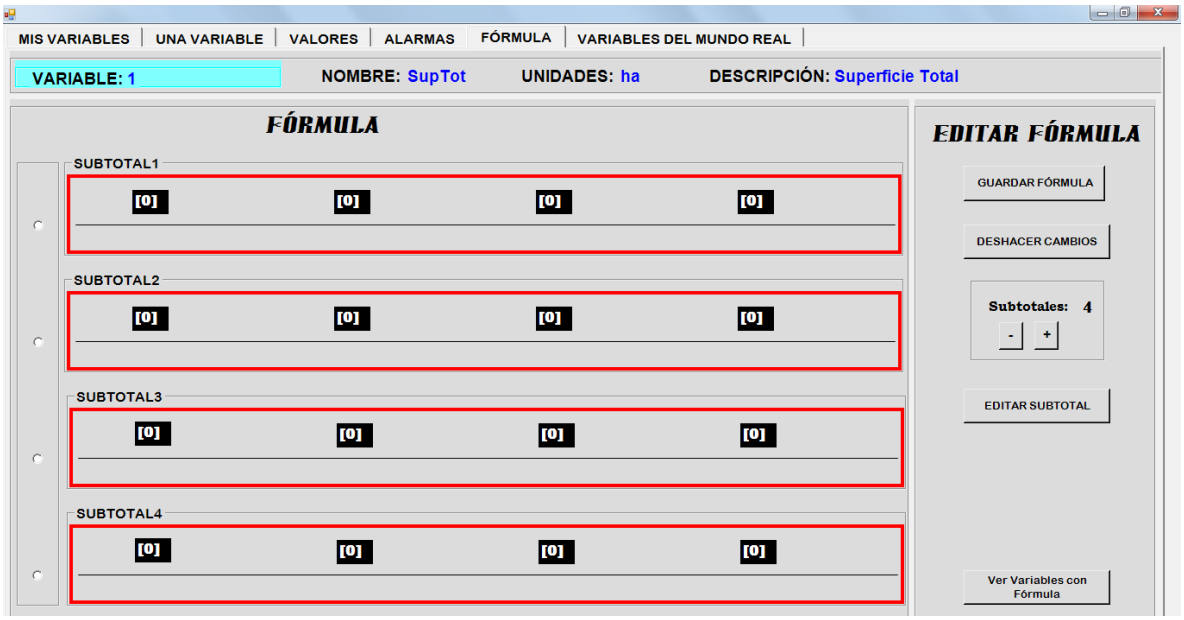


Figura 24. Subtotales agregados

Por ahora no tienen datos, para modificarlos hay que seleccionar un subtotal en la barra de la izquierda, en el *radiobutton* correspondiente y después se clickea el botón "EDITAR SUBTOTAL".

Figura 25. Editar un subtotal

En la Figura 25 se muestran los operandos que tiene el subtotal 1. En el ejemplo aparecen sólo tres pero con las flechas de la caja "Operandos" se pueden agregar más o quitar.

Tomando en cuenta la Figura 25, es decir, la edición de un subtotal, lo siguiente es seleccionar un operando para modificarlo. Para eso hay que clickear el botón “Editar”.

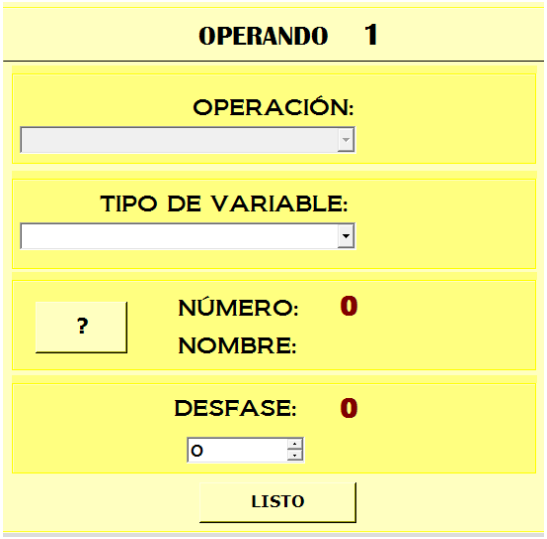


Figura 26. Editar un operando

En la Figura 26 se muestran los atributos del operando que será modificado El título del cuadro muestra el número del operando que está siendo modificado, en este caso es el 1. La opción de la operación está desactivada, ya que antes debe elegir qué tipo de variable es la que se va a usar en este operando.

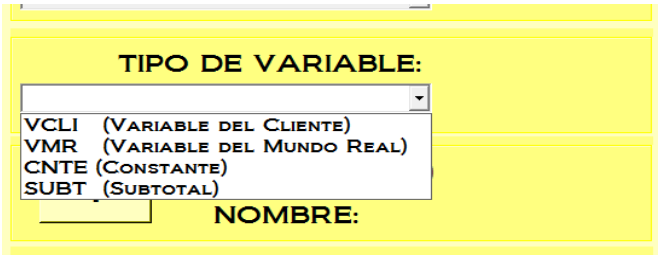


Figura 27. Elegir tipo de variable en operando

Como se puede ver en la Figura 27 hay cuatro opciones. Al seleccionar una, se indica el tipo en la parte derecha.

Figura 28. Tipo de variable seleccionada: VCL

Después de elegir el tipo (ver Figura 28), hay que seleccionar el número. Para eso se presiona el botón “?”.

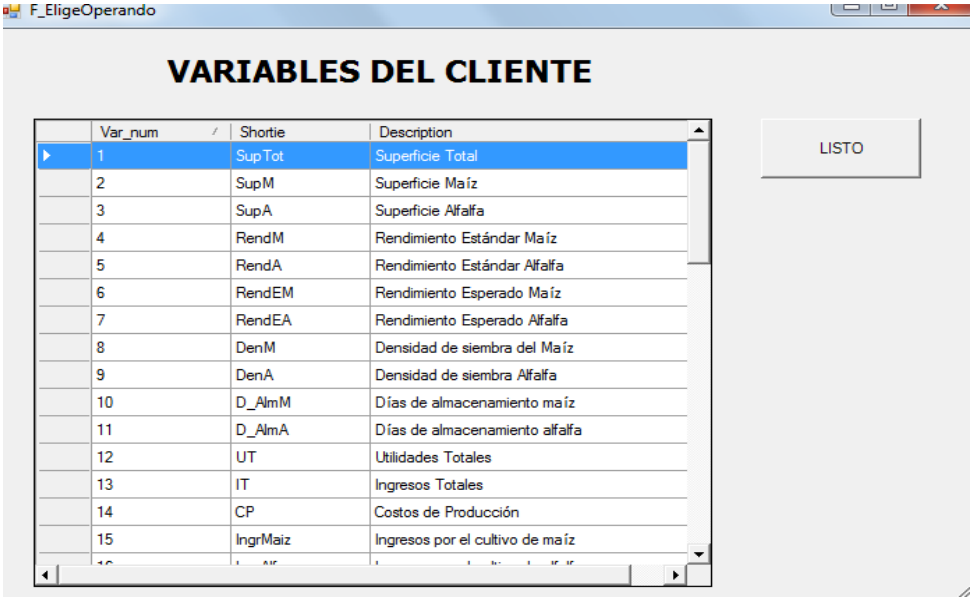


Figura 29. Elegir una variable de cliente

Se muestra un cuadro con todas las variables del cliente que tiene su modelo, de las cuales puede elegir la que desee, aquí se eligió la número 1. La forma mostrada en la Figura 29 representa las opciones a elegir para el operando que se está editando, si el tipo hubiese sido VMR, el cuadro muestra las VMR que hay en el modelo y en el caso de un subtotal, se verán los mismos que estén disponibles.

Figura 30. Variable agregada a operando

Figura 31. Elegir operación inicial para operando escalar

Ahora se elige la operación. Como es el operando inicial y la variable es tipo escalar (ver Figura 30), las operaciones disponibles son: “raíz”, “cuadrado”, “Max”, “Min” y “-”. Esto se puede ver en la Figura 31 conforme a lo mencionado en la Tabla 11.

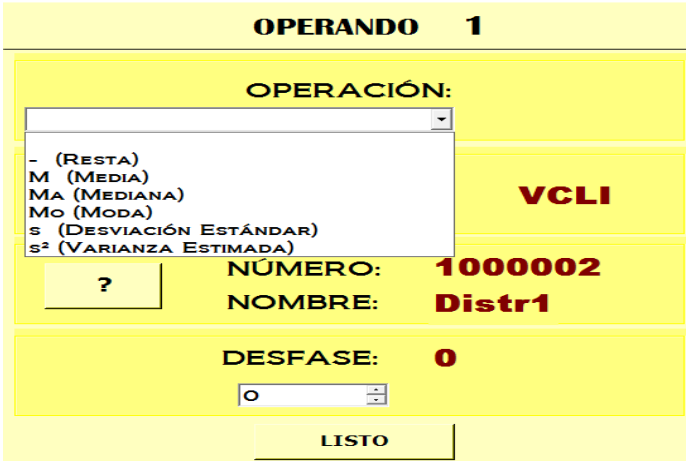


Figura 32. Elegir operación inicial para operando aleatorio

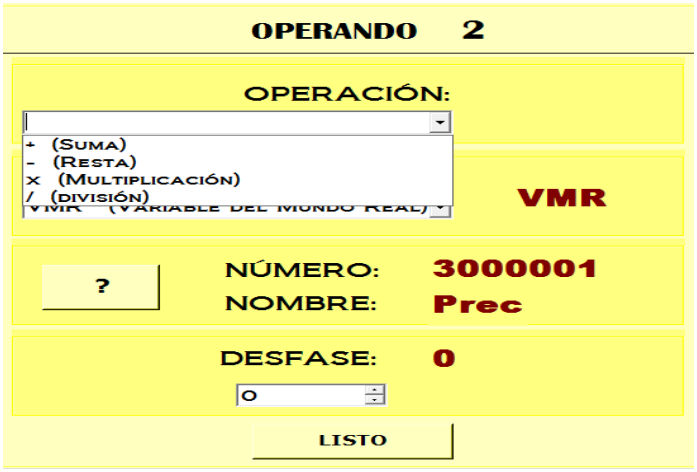


Figura 33. Elegir operación para operando no inicial

Figura 34. Operación elegida

Para la variable tipo distribución se muestran solo las operaciones que le competen (ver Figura 32). Cuando el operando no es el inicial, se muestran las operaciones que se observan en la Figura 33.

Al igual que al elegir el tipo de variable, en la parte derecha aparece la opción seleccionada. En la caja numérica de la parte de abajo está la opción de elegir el desfase, así se muestra en la Figura 34. En Figura 35 se puede ver que ya están los datos del operando 1 agregados al subtotal.

Figura 35. Un operando agregado al subtotal

Figura 36. Un subtotal completo

La Figura 36 muestra el ejemplo de un subtotal con 4 operandos elegidos, entre los que hay variables del mundo real y variables del cliente.

3.4. Valores de las variables del cliente

El cliente tiene que mantener al día los valores de sus propias variables (VCL). Sólo proporciona los valores de las variables de abajo (variables sin fórmula). Para ello, se le proporcionan interfaces para teclear dichos datos, con algunas facilidades para hacerlo en forma más eficiente.

Por la naturaleza de los valores, se proporcionan interfaces separadas para indicar valores de variables aleatorias, dado que en lugar de un valor por período, hay que indicar (hasta) 4 pares de “valor – probabilidad”.

Antes de describir las funciones que se usan para proporcionar los valores, se describen los dos archivos que se usan para almacenarlos. En ambos casos, se trata de archivo planos que se usan con acceso aleatorio (RANDOM ACCESS).

Los valores de cada variable, son vectores de tipo entero y se guardan en dos archivos planos llamados *Vectors.CLIVARS.NUMVALUE.jbr* para valores escalares y *Vectors.CLIVARS.DNVALUE.jbr* para valores de distribución. Se hizo de esta forma porque tienen estructuras diferentes. Por el hecho de que la cantidad de variables no sobrepasarán los 999999 en cada archivo, los registros se enumeran desde 1 a 999999 para ambos casos, escalar y distribución.

Es importante señalar que para estos efectos se usa el número que le asigna el cliente a cada VMR que utiliza. Cómo éstas tendrán números de 2000001 a 999999 para VMR escalares y 3000001 a 3999999 para aleatorias, para grabar y leer el archivo se usa el número de cliente MOD 1000000, lo que equivale a “quitarle los millones” a los números.

3.4.1. Partes que componen la captura de valores

En la Figura 37 se puede ver la interfaz que permite la captura de valores. En la parte de arriba están los datos generales de la variable que fue seleccionada en la pestaña 1. En fondo azul claro está el número de la variable y en seguida el nombre, tipo de periodos, unidades, descripción y factor aplicable. Después se encuentra otro cuadro que contiene los valores actuales de esa variable, primero están los datos del periodo inicial, el año y otras opciones. En la parte de la derecha hay dos botones para guardar lo que está en las cajas de valores o bien restaurar los últimos que se guardaron. El cuadro de la derecha es una herramienta para la captura de valores. Finalmente están las cajas de valores que son 24 ya que es el número máximo de periodos que pueden tener las variables, en caso de que haya algunas que no se utilicen se hace filtro para que no aparezcan ya que en dicho caso el valor default es de 0.

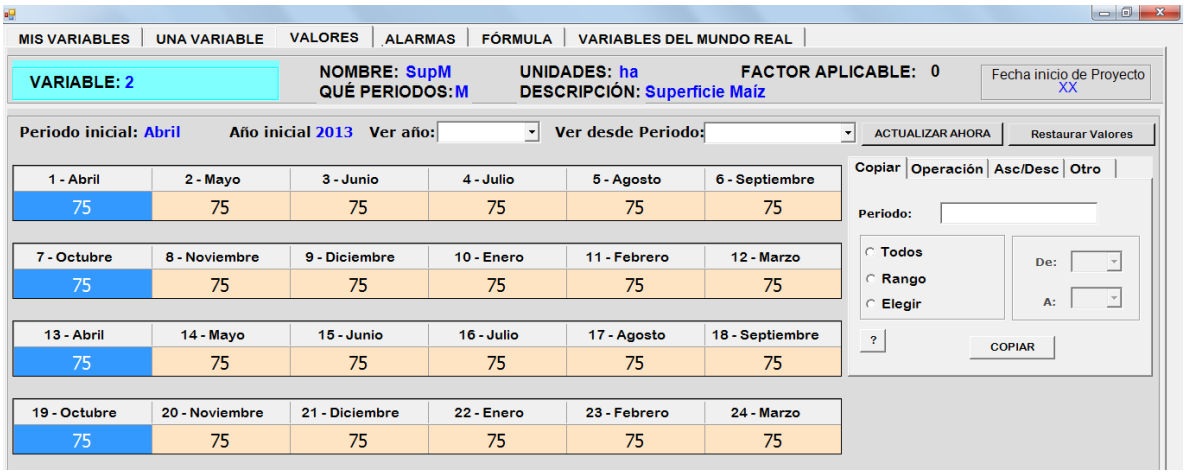


Figura 37. Interfaz de captura de valores escalares

Los valores que se van a guardar son de tipo entero largo, se decidió así porque pueden contener valores de hasta 10 dígitos. La mayoría de variables no tendrán cantidades tan grandes, es por eso que se convino hacerlo así, los modelos de la vida real no usan cantidades tan exageradas y el sistema puede soportar bastante bien los modelos con este tipo de variables. Los valores a guardar siempre serán enteros, sin embargo, la captura es de forma normal, es decir, el cliente la hará con los puntos decimales cuando sea necesario y el programa se encarga de generar el factor aplicable y convertir los valores, de tipo flotante, a enteros y así guardarlos en el archivo. Este proceso se explica más adelante.

3.4.2. Captura

Los valores se pueden teclear en las cajas de cada periodo tal cual los tiene el cliente (como se ve en la Figura 37), sin embargo, existe una herramienta que se puede usar en varios casos y hace que esto sea mucho más eficiente.

Copiar valores: variables escalares

Lo que se debe hacer aquí es teclear el valor en la caja de texto “Periodo”, mostrado en la Figura 38, y seleccionar una de las opciones: todos, rango o elegir. En la opción *Todos*, el valor se copiará a todos los periodos.

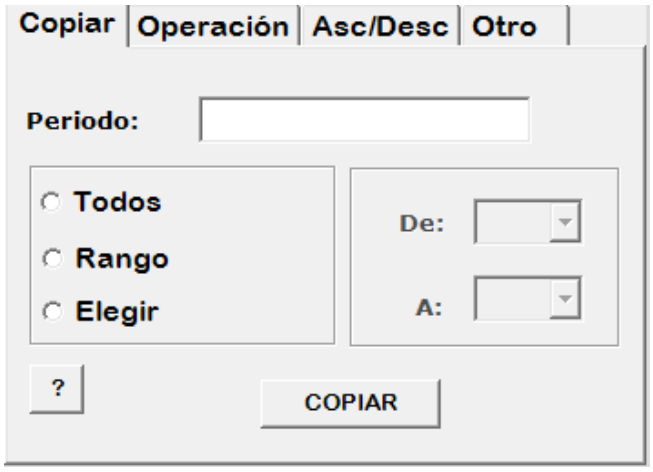


Figura 38. Herramienta de captura de valores (escalares): copiar

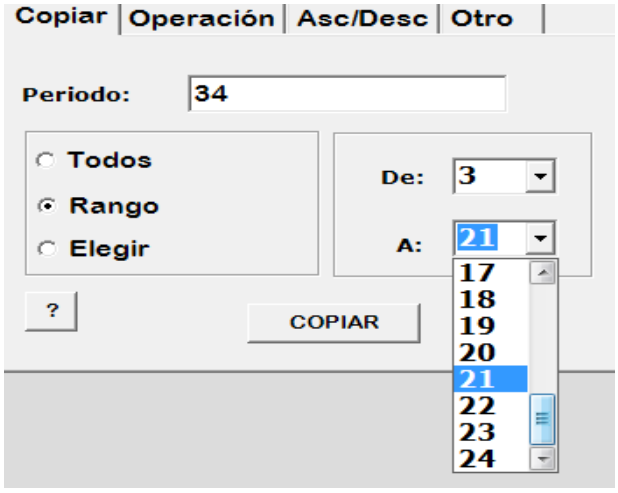


Figura 39. Seleccionar rango para copiar valores

En *Rango* se activarán los combos "*De"* y "*A*" y se elegirá desde qué periodo hasta cual se desea copiar el valor (ver Figura 39).

En la Figura 40 se observa la opción *Elegir*, donde aparecerá una casilla de selección para cada periodo y, de esta manera, el cliente activará los que se desee que tengan dicho valor (ver Figura 41).

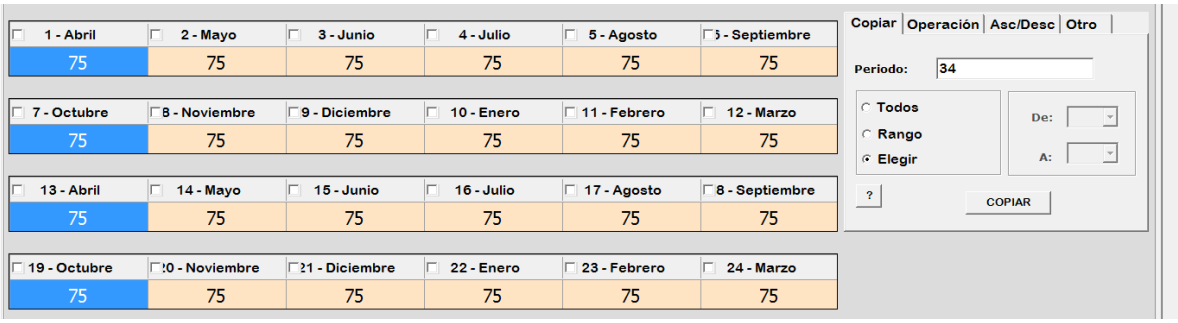


Figura 40. Elegir periodos para copiar valores

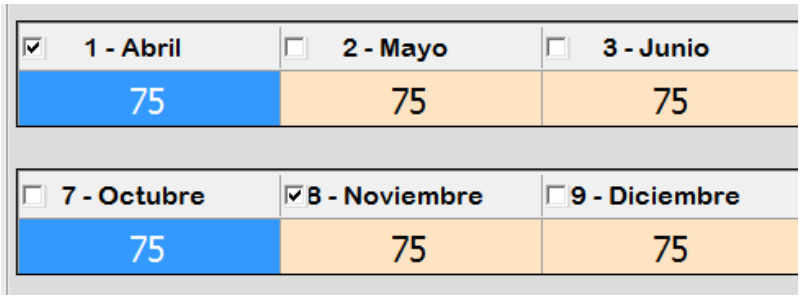


Figura 41. Casillas seleccionadas de algunos periodos

Copiar valores: variables de distribución

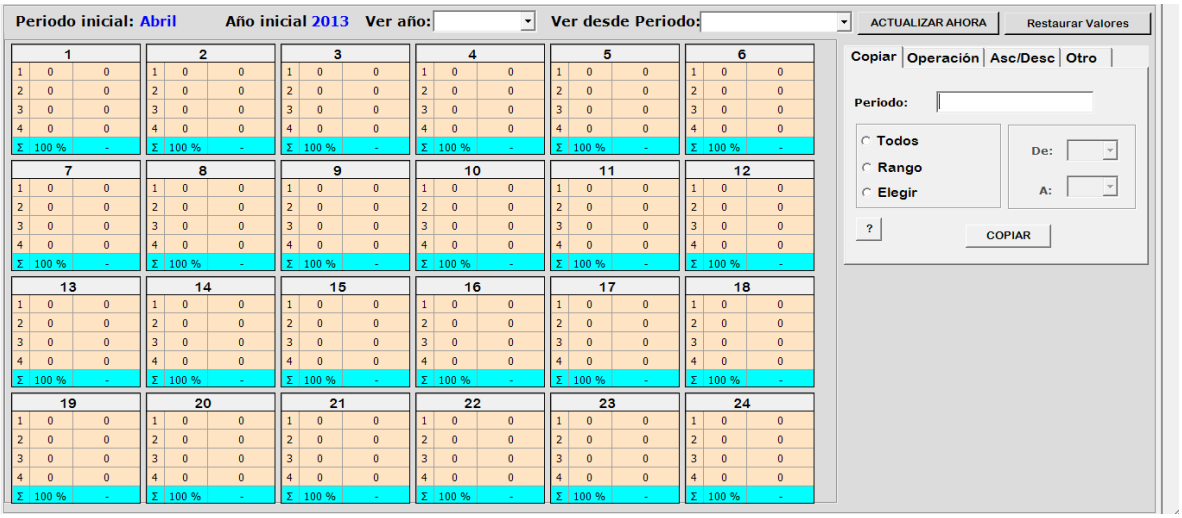


Figura 42. Herramienta de captura de valores (aleatorios): copiar

Para copiar valores en una variable de distribución primero se debe llenar un periodo con valores, para eso primero se selecciona un periodo (ver Figura 42).

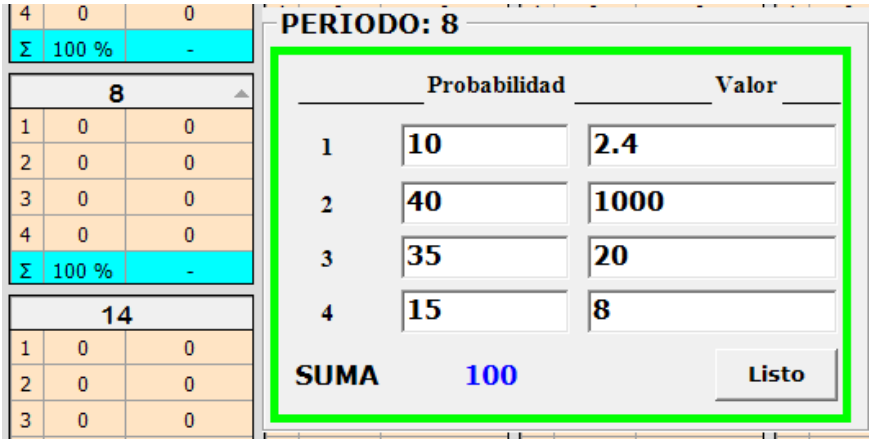


Figura 43. Editar un periodo de una variable aleatoria

La Figura 43 muestra la forma de capturar datos de un periodo en una variable aleatoria. Se pueden ver los pares “valor – probabilidad”. Una restricción que hay

aquí es que la probabilidad siempre debe sumar 100%, por eso el programa no deja actualizar un periodo sino hasta que la suma de las probabilidades sea de 100. Para el caso de valores no hay problema, con la excepción de que los caracteres deben ser numéricos. Con los datos capturados para ese periodo, se da click en “Listo” y queda el periodo como lo muestra la Figura 44.

8		
1	10	2.4
2	40	1000
3	35	20
4	15	8
Σ	100	-

Figura 44. Un periodo aleatorio capturado

Ahora para copiar este periodo, el proceso se hace de forma análoga a como se hizo para una variable escalar. La diferencia es que ahora se tecleará el número del período que se desea copiar y las demás opciones se usan de la misma forma que antes.

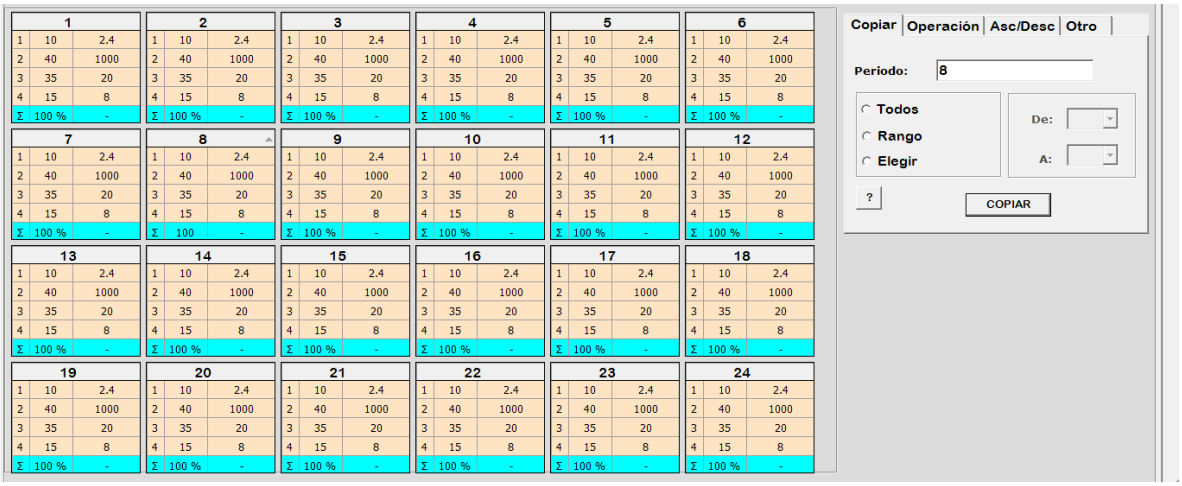


Figura 45. Un periodo aleatorio copiado a todos

Se puede ver en la Figura 45 que se eligió el periodo 8 que ya contenía valores y se copió a todos los demás.

Otras herramientas

En el caso de las herramientas “Operación” y “Asc/Desc” de la Figura 46, al ejecutarse va a tomar de referencia el primer periodo y de ahí empezará a modificar los demás. En la opción *Todos* siempre va a tomar el periodo 1, en *Rango* el que esté en la casilla *De* y en *Elegir* será el primer *checkbox* que esté

seleccionado. Por lo que, necesariamente, ese primer periodo debe ya contener valores.

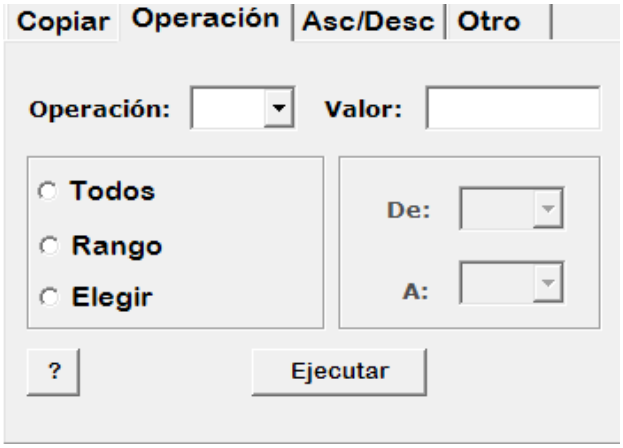


Figura 46. Herramienta “operación” para la captura de valores

Operación: al valor del primer periodo se le aplicará la operación seleccionada aplicada al valor contenido en la caja de texto “Valor” y esto se aplicará a los periodos convenidos por las opciones.

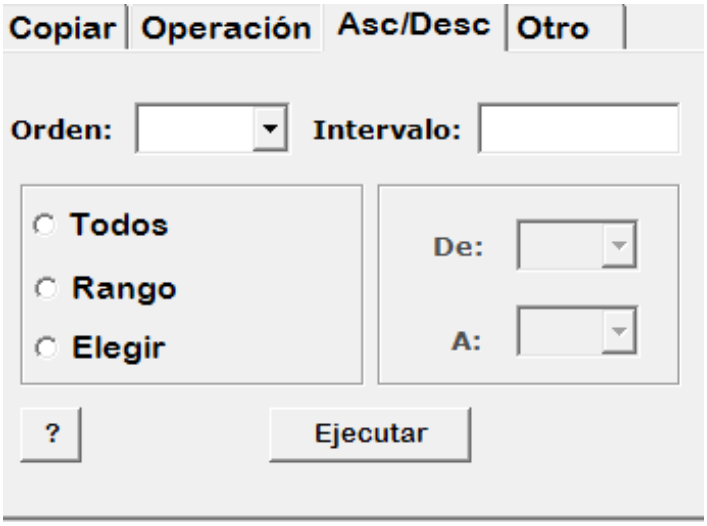


Figura 47. Herramienta “Asc/Desc” para la captura de valores

Asc/Desc: en este caso los valores de los demás periodos irán disminuyendo o aumentando según lo que se haya elegido en la casilla “Orden”. La cantidad que esté en la caja “Intervalo” es lo que se aplicará en el aumento o disminución de los valores para llenar los demás periodos. También puede elegir los periodos a los cuales desea aplicar este proceso, al igual que en los otros casos (ver Figura 47).

3.4.3. Factor Aplicable

Una característica importante de los valores es el factor aplicable ya que éste define la máxima cantidad de decimales que tienen los periodos capturados y así se decide cuál sea el valor entero final que se va a guardar en el sistema. Al introducir un valor, por ejemplo, de 34.567 el factor aplicable es de -3 por los 3 decimales que hay después del punto; otro periodo con valor de 1.43768, tiene factor aplicable de -5. Lo que hace este factor es transformar el número decimal a entero para que pueda ser guardado.

Tabla 12. Ajuste del factor aplicable en una variable

Periodo	Valor real	Valor en disco	Factor aplicable
1	3.45	3450000	-6
2	5.6789	5678900	-6
3	1.000045	1000045	-6
4	1034.921	1034921000	-6
…			
24	88	88000000	-6

Un factor aplicable es el mismo para todos los periodos de una variable independientemente de los decimales que tenga cada uno. Esto se ve claramente en la Tabla 12 en la que el factor aplicable es de -6 y se aplica a cada periodo. En la interfaz del programa se introduce 3.45 pero se guarda 3450000 porque el periodo con mayor número de decimales es el 3 que tiene 6. El procedimiento que se sigue es que cuando se confirma la actualización de los valores, una rutina recorre todos los periodos y determina el que tiene más decimales; ése es el factor aplicable de la variable. En un arreglo de 24 valores se almacenan los valores originales pero multiplicados por 10^6 (el 6 es el factor) y ese vector es el que se guarda en el archivo. Para extraer los datos del archivo, éstos se multiplican por 10^{-6}; de esta forma se muestran como normalmente se capturaron ya que los ceros a la derecha desaparecen.

Para guardar los valores en cada archivo se siguen los siguientes pasos:

- Primero se introducen los 24 valores, como se explicó antes;
- Después, al “Actualizar valores”, se llama a la rutina que determina el factor aplicable y genera el arreglo en memoria de los valores enteros finales;
- Hecho esto se abre el archivo correspondiente donde se van a grabar los datos. Recordar que el número de la variable dice si es escalar o distribución.
- Se ubica la posición del registro donde se va guardar la estructura que está en memoria, tal posición es el número de la variable. Para el caso de las distribuciones se aplica módulo 1000000.
- Se graba y cierra el archivo correspondiente.

3.5. Criterios de ejecución

El llamado “motor” del FLAG consiste esencialmente de dos procesos: la consecución de valores actualizados de las VMR que ofrece, y la ejecución de los modelos afectados por dichos cambios. Los clientes pueden solicitar que se ejecuten sus modelos cuando algunas de las VMR sufran cambios *significativos,* cuestión que ellos mismos deben definir. Para esto, formulan criterios de ejecución de sus modelos, los cuales reflejan lo que para ellos (los clientes) son las variaciones que tienen un impacto suficiente sobre sus indicadores para que amerite una ejecución del modelo. Aunque – como se ha dicho previamente - en esta versión del FLAG no se ha incluido el módulo de “contabilidad”, se supone que la consecución de valores y las ejecuciones de modelos tienen un costo para el cliente.

3.5.1. Definición de los criterios

Los criterios de ejecución de su modelo se formulan para una o más variables del mundo real del modelo. Para cada una de ellas, se indica un rango de variación, de modo que si el valor de la VMR cambia más que un cierto porcentaje (o una cantidad indicada para tal efecto) ya sea en sentido positivo o negativo, se ejecutará el modelo del cliente. En otras palabras, se define un intervalo que contiene al valor actual de ese período de la variable, y si el nuevo valor cayera fuera de dicho intervalo, se ejecuta el modelo.

Tabla 13. Reglas para criterios de ejecución

REGLA	SE COMPARA CON	TIPO DE VARIACIÓN
1	Agente	Escalar
2	Valor propio	Escalar
3	Agente	Porcentual
4	Valor propio	Porcentual

Los criterios de ejecución se dividen en dos componentes importantes: la regla y los límites. En la Tabla 13 se muestran las reglas que definirán las características del criterio que elegirá un cliente para una VMR. El tipo de variación es el valor de los límites definidos, que pueden estar indicados por una cantidad constante, mayor o menor a la anterior valor de la variable, o puede ser un cierto porcentaje del mismo. El “valor propio” se refiere al último valor del cliente que provocó una ejecución, es decir, el último valor que superó los límites establecidos en el criterio. Y cuando se compara con el valor del “Agente” se refiere al valor anterior que se obtuvo del mismo, de acuerdo a su periodicidad mínima. Por ejemplo, si esta VMR tiene como periodicidad de actualización por el agente un minuto, cada vez obtiene un nuevo valor, si en el criterio se eligió la regla 1 o 3, el programa estará comparando siempre con el valor del minuto anterior.

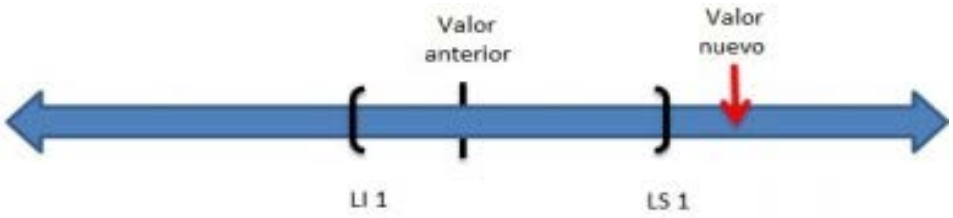


Figura 48. Límites para criterios de ejecución

Los límites de un criterio se definen alrededor del valor anterior de la variable, que siempre es conocido. Cuando el valor nuevo cae fuera del rango de incidencia (ver Figura 48), se dispara la ejecución para este cliente. En el caso contrario, no procede nada. Se mencionó, en la reglas, que los límites son de dos tipos, escalares y porcentuales; en el primer caso si, por ejemplo, se introduce como límite superior el valor de 12, el criterio lo leerá como doce unidades mayor al valor anterior; en cambio, si la regla es de tipo porcentual, el límite superior es igual a 12% por encima del valor anterior.

Tabla 14. Estructura de la tabla “criterios de ejecución” de la base de datos

CAMPO	TIPÓ	DESCRIPCIÓN
Número de variable	Entero	Variable del mundo real a la que se le define criterio
Regla usada	Byte	Número de la regla
Límite inferior	Flotante	Valor mínimo del rango para concretar la ejecución
Límite superior	Flotante	Valor máximo del rango para concretar la ejecución

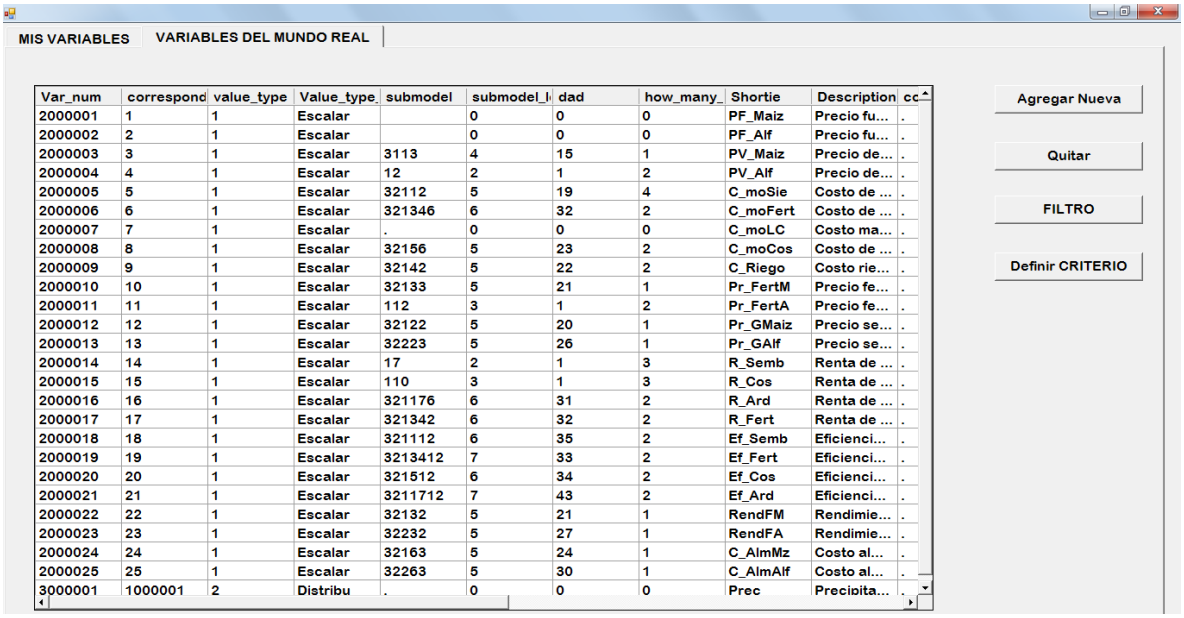
En la Tabla 14 se muestra la información que se guarda en la base de datos. Cada registro de la tabla es un criterio definido para una y sólo una VMR, lo que quiere decir que el número de la variable es el campo llave de la tabla; no se puede definir más de un criterio por cada VMR.

Comentario: el motor del FLAG no usa los datos directamente de la Tabla 14 en la base de datos, puesto que para hacerlo constantemente tendría que leer los modelos de todos los clientes, cada vez que se detecte un cambio de valor de una VMR. FLAG evita esto en dos pasos: se construye un archivo que contiene los criterios de ejecución de todos los clientes, y el motor carga estos datos a memoria para mayor eficiencia del proceso de decisión que conduce a la ejecución de los modelos. Esto se describe a mayor detalle en el capítulo 6.

3.5.2. Captura de criterios

La Figura 49 muestra la interfaz para indicar los criterios de ejecución de variables del mundo real. En la parte de la derecha, aparecen cuatro botones: agregar nueva variable, quitar, filtrar y definir criterio. Para definir el criterio de alguna

variable, primero debe seleccionarse del cuadro de variables y después presionar el botón “definir criterio”



Var_num	correspond	value_type	Value_type	submodel	submodel_l	dad	how_many_	Shortie	Description	cc
2000001	1	1	Escalar		0	0	0	PF_Maiz	Precio fu...	.
2000002	2	1	Escalar		0	0	0	PF_Alf	Precio fu...	.
2000003	3	1	Escalar	3113	4	15	1	PV_Maiz	Precio de...	.
2000004	4	1	Escalar	12	2	1	2	PV_Alf	Precio de...	.
2000005	5	1	Escalar	32112	5	19	4	C_moSie	Costo de ...	.
2000006	6	1	Escalar	321346	6	32	2	C_moFert	Costo de ...	.
2000007	7	1	Escalar	.	0	0	0	C_moLC	Costo ma...	.
2000008	8	1	Escalar	32156	5	23	2	C_moCos	Costo de ...	.
2000009	9	1	Escalar	32142	5	22	2	C_Riego	Costo rie...	.
2000010	10	1	Escalar	32133	5	21	1	Pr_FertM	Precio fe...	.
2000011	11	1	Escalar	112	3	1	2	Pr_FertA	Precio fe...	.
2000012	12	1	Escalar	32122	5	20	1	Pr_GMaiz	Precio se...	.
2000013	13	1	Escalar	32223	5	26	1	Pr_GAlf	Precio se...	.
2000014	14	1	Escalar	17	2	1	3	R_Semb	Renta de ...	.
2000015	15	1	Escalar	110	3	1	3	R_Cos	Renta de ...	.
2000016	16	1	Escalar	321176	6	31	2	R_Ard	Renta de ...	.
2000017	17	1	Escalar	321342	6	32	2	R_Fert	Renta de ...	.
2000018	18	1	Escalar	321112	6	35	2	Ef_Semb	Eficienci...	.
2000019	19	1	Escalar	3213412	7	33	2	Ef_Fert	Eficienci...	.
2000020	20	1	Escalar	321512	6	34	2	Ef_Cos	Eficienci...	.
2000021	21	1	Escalar	3211712	7	43	2	Ef_Ard	Eficienci...	.
2000022	22	1	Escalar	32132	5	21	1	RendFM	Rendimie...	.
2000023	23	1	Escalar	32232	5	27	1	RendFA	Rendimie...	.
2000024	24	1	Escalar	32163	5	24	1	C_AlmMz	Costo al...	.
2000025	25	1	Escalar	32263	5	30	1	C_AlmAlf	Costo al...	.
3000001	1000001	2	Distribu	.	0	0	0	Prec	Precipita...	.

Figura 49. Interfaz de variables del mundo real

El criterio se define en un cuadro como el que se ilustra mediante la Figura 50, donde hay que elegir la regla y los límites. Se dedicó un esfuerzo considerable tanto en el diseño como la programación de este elemento, puesto que se trata de ayudar a los clientes a hacerlo correctamente. El artificio de mostrar el intervalo que están construyendo aparentemente simplificará su uso.

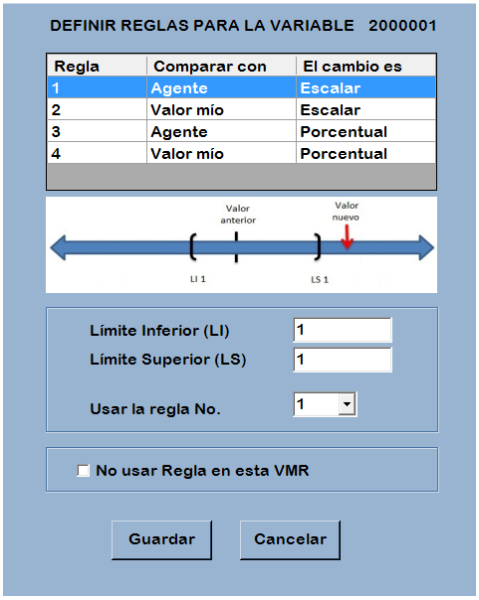


Figura 50. Ejemplo de un criterio de ejecución definido

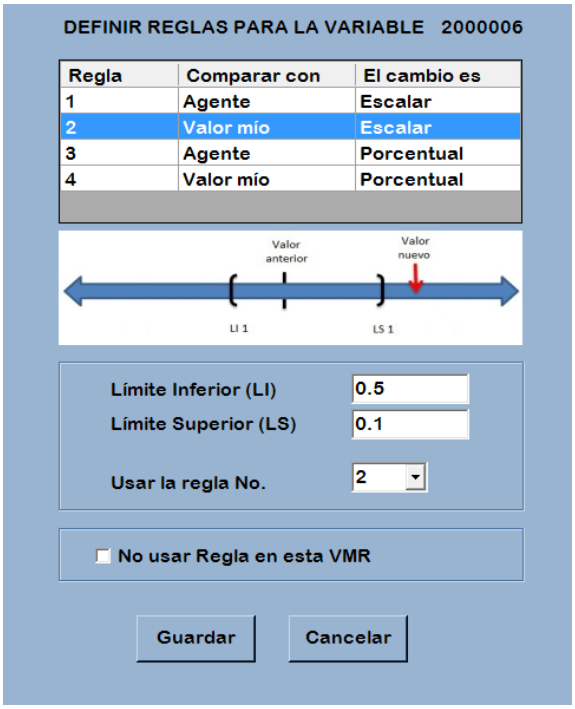


Figura 51. Ejemplo de otro criterio de ejecución definido

En el ejemplo de la Figura 51 se está usando la regla número 2, lo que quiere decir que se usa el último valor que hizo que cambiara el modelo y que si el valor nuevo es menor que el anterior menos 0.5 o mayor al anterior más 0.1, este criterio se aplica y el submodelo se debe ejecutar.

3.6. Criterios de alarma

Un cliente puede solicitar que se le emita un aviso urgente (en FLAG se llaman *alarmas*, tras considerar otros nombres) cuando alguno de sus indicadores (algunas variables del cliente del modelo) sufran cambios significativos en cualquier sentido. La RAE define el término alarma como “mecanismo que, por diversos procedimientos, tiene por función avisar de algo”. Hay otras definiciones que enfatizan algún aspecto de “urgencia”, y en este trabajo el uso de ese vocablo refleja precisamente eso: que se necesita advertir a un cliente de algo que él indicó que era significativo para su negocio o actividad.

La selección del término alarma resultó de un análisis de varias denominaciones posibles: alertas, avisos, advertencias, mensajes urgentes, etc. A la larga, se decidió llamar alarmas a estos mensajes porque en cierto modo, su significado abarca los de todos los términos alternos.

De ese modo, una alarma es un aviso a un cliente de FLAG en el sentido de que se cumplió algo que el cliente considera “suficientemente significativo para merecer un tal aviso”.

Para ello, selecciona sus variables *críticas,* que son a las que les agregará un criterio de alarma. Cuando la ejecución de su modelo resulte en que se cumpla el criterio de alarma de una variable crítica, se acumulará un puntaje a la gravedad de la ejecución. Si la gravedad cae en los rangos definidos por el cliente, se le enviará un mensaje.

Para indicar esto último, el cliente formula “criterios de alarma”: condiciones que se presentan que, a su vez, hacen que algunas de sus variables sufran modificaciones significativas donde, naturalmente, este atributo depende totalmente de su modelo y del uso que se le dé.

Resumiendo: el cliente formula los denominados criterios de alarma; cuando, como consecuencia del recálculo de variables en la ejecución de su modelo, los cambios satisfagan esos criterios, FLAG le enviará al cliente una alarma, en la modalidad que seleccionó e indicó como parte de los datos que proporciona al FLAG.

En un trabajo de Tesis anterior, Gilberto Velázquez (2009) desarrolló los criterios de alarma, su aplicación durante la ejecución de modelos y los aspectos del envío de las alarmas al cliente. En esta nueva versión del FLAG, se mantuvieron todos los elementos definidos en aquella ocasión, pero se modificaron las estructuras de datos que usa el programa de ejecución para hacerlo más eficiente. También se reprogramaron con el mismo objetivo las rutinas que aplican los criterios a los cambios en los valores de las variables. Se puede afirmar que de todos los componentes del FLAG éste es el que tuvo menos modificaciones, a pesar de que se cambiaron las interfaces para especificar las alarmas por parte de un usuario, puesto que las anteriores eran complicadas y confusas, lo que resultó en que se decidió hacerlas más intuitivas, esto siempre pensando en la comodidad del cliente.

En este capítulo sólo se describen los criterios de alarma. La comprobación de que se produjo una situación que merece una alarma y el envío de la misma al cliente se describirán en secciones posteriores.

3.6.1. Descripción de los criterios de alarma

Un criterio de alarma se “coloca” (define para) sobre una variable del cliente, que de ese modo merece el calificativo de *variable crítica*. Necesariamente se trata de una variable CALCULADA. No se pueden definir alarmas para VMR. Si un cliente quisiera hacerlo, solamente tiene que introducir una variable (VCL) en cuya fórmula aparezca solamente la VMR que desea.

Adicionalmente, el FLAG no permite definir criterios de alarma para variables aleatorias. Una vez más, si el cliente tiene necesidad de que le avisen cuando

cambie significativamente una variable, podrá usar una estadística (por ejemplo la media o moda): define una VCL escalar = media (VCL–aleatoria) y especifica el criterio de alarma sobre la variable escalar calculada.

Un criterio de alarma consta de 4 elementos principales:

- La variable a la cual se refiere
- Los periodos a los que se aplica el criterio
- Los intervalos críticos alrededor del valor (“anterior”)
- Un *índice de gravedad* del cambio.

En la ejecución de un modelo, se acumulan las gravedades encontradas como resultado de los cambios de los valores de las variables.

El cliente, como parte de las especificaciones de sus necesidades, especifica rangos de esta gravedad que determinarán el tipo de alarma que se le enviará. El FLAG usa los 4 rangos ilustrados en la Tabla 15 para especificar los 4 tipos (o modos) de alarma que se pueden invocar.

Tabla 15. Modos de alarma en FLAG

MODO	MEDIO DE ENVÍO	DESCRIPCIÓN DEL MENSAJE
0	No hay alarma	No se genera mensaje
1	E – mail	Se avisa que el modelo ha cambiado
2	E – mail	El modeló cambió significativamente. Se menciona la variable, el periodo y la cantidad que cambió
3	SMS	El mensaje es el mismo que en el modo 2, solo que como la gravedad es mayor, se envía por SMS
4	Llamada telefónica	El mensaje se comunica al cliente por medio de una llamada telefónica.

Estos avisos se generan automáticamente con el programa a excepción del modo 4 que se trata de una llamada telefónica que, hasta ahora, es necesario que se haga personalmente por un usuario de FLAG. De hecho, lo llama un especialista del FLAG y le comenta los cambios en su modelo, para lo cual “carga” ese modelo y usa consultas para determinar ciertos datos. En un momento dado le podría aconsejar alguna acción, si el tema del modelo fuera parte de su especialidad.

Naturalmente en una instancia de FLAG, estas alarmas probablemente implicarían algún costo para el cliente, que se incrementaría de acuerdo al tipo de alarma que se genera. Cuesta más enviar un SMS que un correo electrónico (que casi no tiene costo excepto el de los procesos involucrados). El tiempo invertido en hacer una llamada personal probalemetne sería más caro aún.

3.6.2. Descripción del modelo de datos

Los criterios de alarmas se almacenan inicialmente en la base de datos del cliente. La información se divide en dos tablas: *alarmas_ variables* y *alarmas_variable_ periodo*. A continuación se detallan y explican cada una de ellas junto con los campos que las componen y la función de cada uno.

Tabla 16. Tabla de alarmas_variables.

CAMPO	TIPO	DESCRIPCIÓN
Número de variable	Entero	Variable del cliente crítica con criterio de alarma definido
Tipo de periodo	Byte	Modalidad de periodos. Define el tipo de selección de periodos que tienen definido un criterio de alarma
Tipo de límites	Byte	Modalidad de comparación de los límites. Son tres tipos: porcentaje, amplitud escalar y valor absoluto.

La llave de la Tabla 16 es el número de la variable ya que sólo es posible asignarle un criterio de alarma a cada variable.

Tabla 17. Tabla de alarmas_variable_periodos.

CAMPO	TIPO	DESCRIPCIÓN
Número de variable	Entero	Variable del cliente crítica con criterio de alarma definido
Periodo inicial	Byte	Primer periodo con criterio. Depende de la modalidad del periodo, definida en la Tabla 16.
Periodo final	Byte	Último periodo con criterio
Usa rango de emergencia	Boolean	Define si el criterio hace uso de los límites de emergencia
Límite crítico inferior	Flotante	Valor inferior del rango crítico
Límite crítico superior	Flotante	Valor superior del rango crítico
Gravedad crítica	Entero	Puntos que acumula el periodo si sobrepasa el rango crítico
Límite emergencia inferior	Flotante	Valor inferior del rango de emergencia
Límite emergencia superior	Flotante	Valor superior del rango de emergencia
Gravedad de emergencia	Entero	Puntos que acumula el periodo si sobrepasa el rango de emergencia

En la tabla de la base de datos ilustrada en la Tabla 17 se almacenan los periodos a los cuales le asignó criterio y los límites de los mismos. En esta tabla la llave es el número de variable junto con el periodo al que se aplica el criterio, que aquí se muestra como periodo inicial. Esto porque una variable tiene hasta 24 periodos y, si es el caso, puede asignarle un criterio a cada uno de ellos. Dependiendo de la modalidad del periodo, guardado en la Tabla 16, es la forma en que será guardado o guardados los criterios de los periodos en la Tabla 17. A

continuación se explica la forma en que se almacenan los registros para cada modalidad.

Tipo 1: un solo periodo con criterio. El campo *periodo inicial* y *periodo final* valen lo mismo. Registros que se guardan por variable: uno.

Tipo 2: todos los periodos. El valor default que se almacena en *periodo inicial* y *final* es 1; aunque es irrelevante ya que se sabe que el criterio guardado es el mismo y será válido para los 24 periodos. También se guarda solo un registro.

Tipo 3: rango de periodos. Los *periodos inicial y final* sí son los que el cliente haya elegido y el criterio es el mismo para los comprendidos en el rango. También es un solo registro.

Tipo 4: lista de periodos. Cuando se elige una lista se seleccionan los periodos a los cuales se les dará un criterio diferente. Por esto es por lo que los registros que se almacenarán en la base de datos es el número de periodos con criterio. El campo *periodo inicial y periodo final* valen lo mismo en cada registro. Se guarda un máximo de 24 registros por variable.

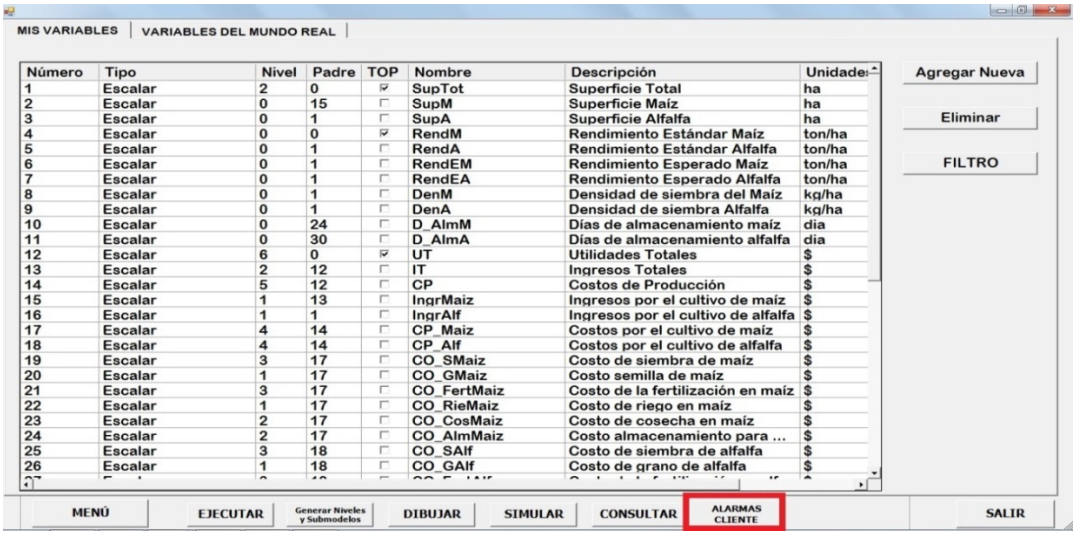
Tipo 5: periodo actual. Si el tipo de periodo de una variable es, por ejemplo, los meses del año y hay valores almacenados desde abril del año en curso y estamos en agosto, entonces la actualización de esta variable específica se hará sobre el periodo 5 que corresponde al mes de agosto. Cuando llegue el mes de septiembre el periodo que se actualizará será el número 6, pero en estos momentos el periodo actual es el 5. Para este tipo de periodo se almacena un solo registro.

Tipo 6: rango de periodos, relativo al actual. Ejemplo, un cliente quiere definir un criterio en el rango [-2, 5], esto quiere decir que el número de periodos afectados es de 8 y si estamos en el periodo 5, el criterio se aplicará desde junio (tomando el ejemplo anterior) del año en curso hasta enero de año siguiente. Se guarda un solo registro, el criterio aplica a todo el rango.

- *Nota sobre los rangos*: en el caso del tipo de periodo 3 está claro que el número que se guarda en el registro es el periodo inicial y final que haya elegido el cliente. En cambio, debido a que el tipo de valor que se usa para el campo *periodo inicial* y *final* es byte (es decir, no acepta números negativos), en el tipo 6 se hace una conversión del número negativo inicialmente elegido por el cliente, esto en caso de que el periodo inicial esté por debajo del periodo actual. Así por ejemplo, si el cliente decide iniciar el rango en -2, el valor que se almacena en el campo de la base es 52, si es -11 se guarda el número 61 y así para cada uno de los periodos que estén antes del periodo actual.

Tipo 7: rango de periodos, relativo al actual. En esta modalidad se elige un rango al igual que en el caso anterior, es decir, con referencia al valor actual; la diferencia es que cada uno de los periodos se maneja de forma independiente, como si fuera una lista, es decir, tienen criterios diferentes. El número de registros que se almacenan es el número de periodos que se incluyen en el rango.

3.6.3. Captura de criterios



Número	Tipo	Nivel	Padre	TOP	Nombre	Descripción	Unidades
1	Escalar	2	0	☑	SupTot	Superficie Total	ha
2	Escalar	0	15	☐	SupM	Superficie Maíz	ha
3	Escalar	0	1	☐	SupA	Superficie Alfalfa	ha
4	Escalar	0	0	☑	RendM	Rendimiento Estándar Maíz	ton/ha
5	Escalar	0	1	☐	RendA	Rendimiento Estándar Alfalfa	ton/ha
6	Escalar	0	1	☐	RendEM	Rendimiento Esperado Maíz	ton/ha
7	Escalar	0	1	☐	RendEA	Rendimiento Esperado Alfalfa	ton/ha
8	Escalar	0	1	☐	DenM	Densidad de siembra del Maíz	kg/ha
9	Escalar	0	1	☐	DenA	Densidad de siembra Alfalfa	kg/ha
10	Escalar	0	24	☐	D_AlmM	Días de almacenamiento maíz	dia
11	Escalar	0	30	☐	D_AlmA	Días de almacenamiento alfalfa	dia
12	Escalar	6	0	☑	UT	Utilidades Totales	$
13	Escalar	2	12	☐	IT	Ingresos Totales	$
14	Escalar	5	12	☐	CP	Costos de Producción	$
15	Escalar	1	13	☐	IngrMaiz	Ingresos por el cultivo de maíz	$
16	Escalar	1	1	☐	IngrAlf	Ingresos por el cultivo de alfalfa	$
17	Escalar	4	14	☐	CP_Maiz	Costos por el cultivo de maíz	$
18	Escalar	4	14	☐	CP_Alf	Costos por el cultivo de alfalfa	$
19	Escalar	3	17	☐	CO_SMaiz	Costo de siembra de maíz	$
20	Escalar	1	17	☐	CO_GMaiz	Costo semilla de maíz	$
21	Escalar	3	17	☐	CO_FertMaiz	Costo de la fertilización en maíz	$
22	Escalar	1	17	☐	CO_RieMaiz	Costo de riego en maíz	$
23	Escalar	2	17	☐	CO_CosMaiz	Costo de cosecha en maíz	$
24	Escalar	2	17	☐	CO_AlmMaiz	Costo almacenamiento para ...	$
25	Escalar	3	18	☐	CO_SAlf	Costo de siembra de alfalfa	$
26	Escalar	1	18	☐	CO_GAlf	Costo de grano de alfalfa	$

Figura 52. Interfaz principal de FLAG – Cliente

Primero, en la interfaz principal se selecciona “Alarmas Cliente” como se ve en la Figura 52. Entonces se abrirá el programa principal de alarmas.

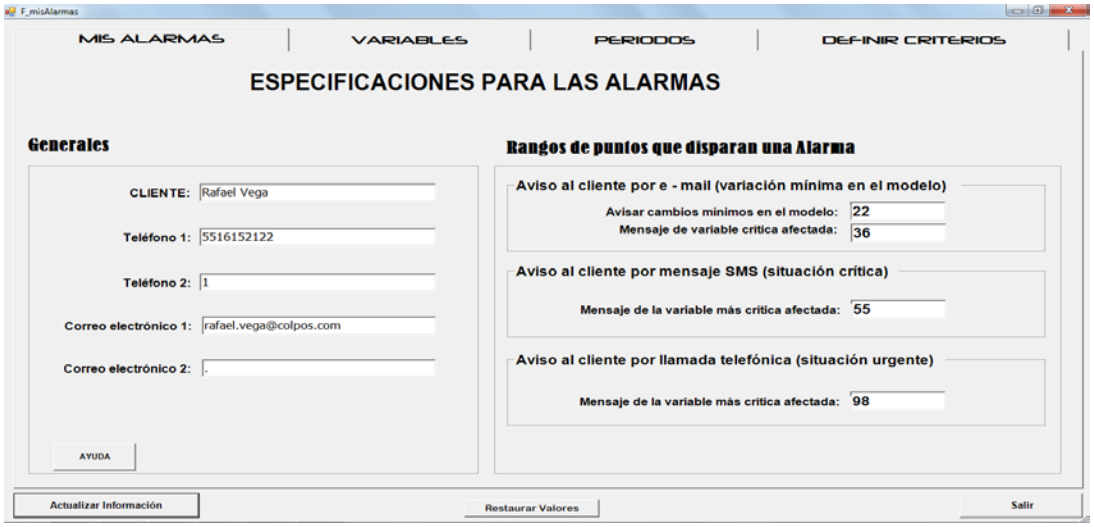


Figura 53. Especificaciones generales de alarmas

En la Figura 53 se muestra la interfaz para los datos principales del cliente como los teléfonos a los que FLAG acudirá en caso de llamada o un SMS, correo o correos electrónicos para enviar también un aviso.

Para los rangos de puntos, el cliente debe ser cuidadoso en el número de puntos que va a asignar para que se le mande cada tipo de aviso, esto porque en caso de poner una cantidad muy alta puede ocurrir que nunca se le avise si los valores de las variables *en general* sufren cambios menores a los indicados. Simplemente se debe ser congruente a la hora de asignar puntajes ya sea para los avisos como para las variables y sus periodos.

Figura 54. Tabulador alarmas: variables

Acto seguido se indican las variables a las cuales asignará criterios, para lo cual se debe seleccionar la segunda pestaña del tabulador: variables (ver Figura 54).

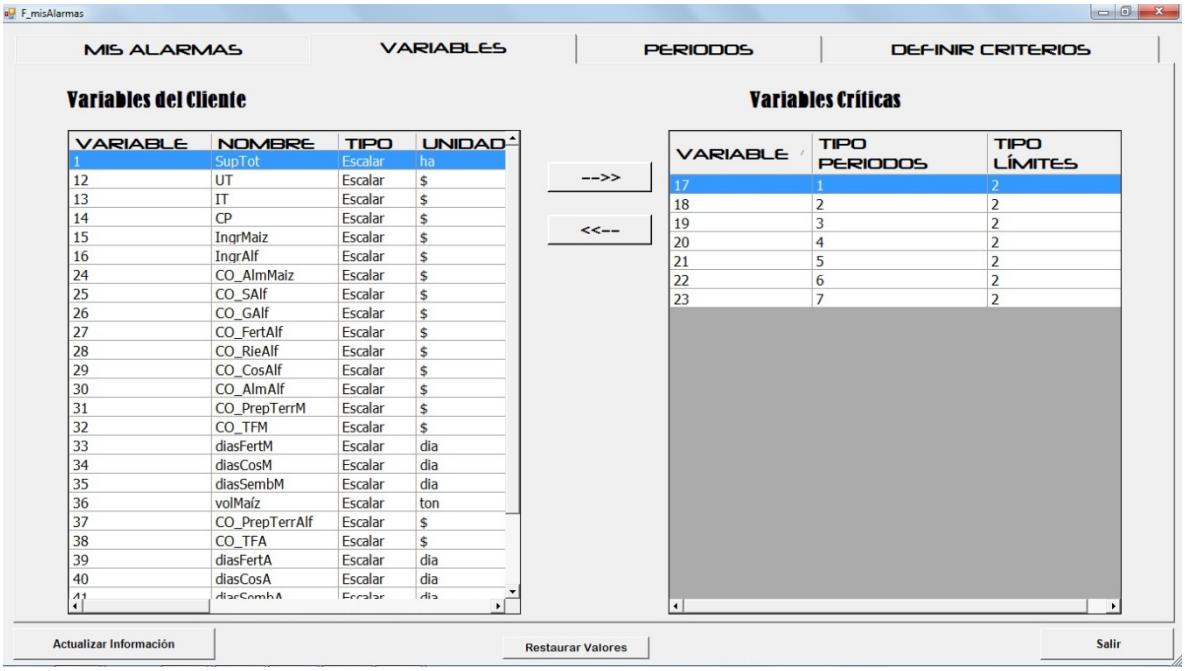


Figura 55. Selección de variables críticas

En la Figura 55 se pueden ver dos cuadros de información y al centro dos botones. El de la izquierda muestra las variables del cliente que puede elegir para asignarles una alarma, es decir, están las variables que tienen nivel mayor a cero (las calculadas) y que no son aleatorias. En la derecha están las variables elegidas para una alarma. Los dos botones son los que envían las variables de un lado para otro, se puede ver que uno tiene símbolos de flecha a la derecha y otro de flecha a la izquierda.

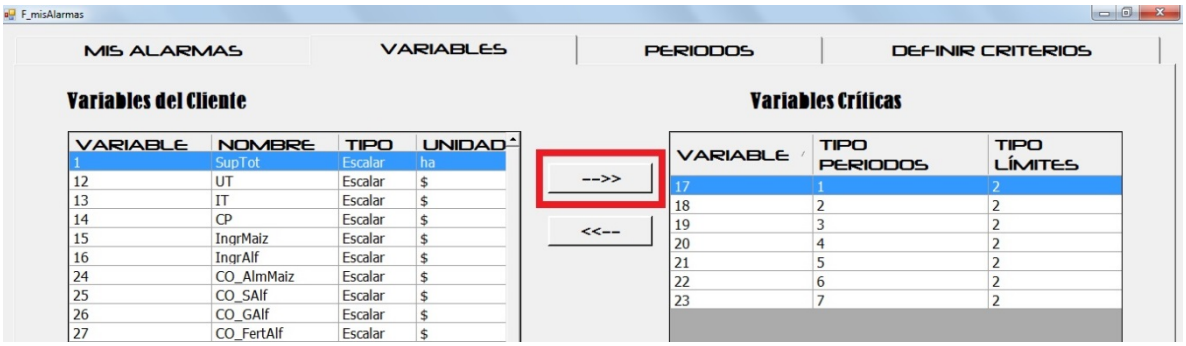


Figura 56. Definir variable como crítica

Cuando se selecciona una variable de cliente y se hace click en *flecha derecha* (ver Figura 56), se manda esa variable al cuadro de variables críticas.

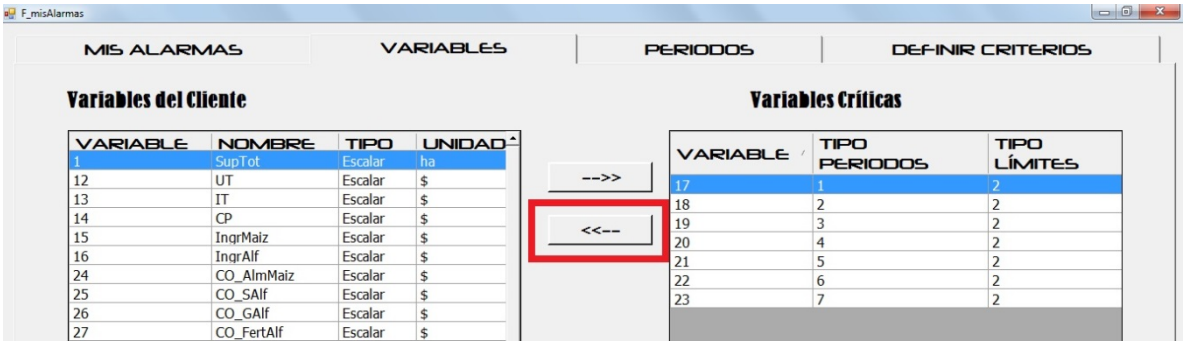


Figura 57. Eliminar variable crítica

Análogamente, ocurre lo mismo al seleccionar una variable crítica y hacer *click* en *fecha izquierda* (ver Figura 57).

Figura 58. Tabulador alarmas: periodos

Se selecciona la pestaña etiquetada como “periodos” para asignar tipo de periodos y tipo de límites en dicha variable (ver Figura 58).

Figura 59. Tabulador periodos vacío

Si no se ha elegido una variable crítica en el cuadro correspondiente o si la variable seleccionada aún no tiene criterio asignado, se verá como la Figura 59.

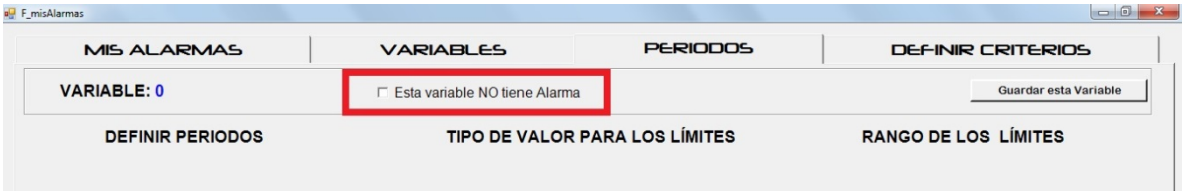


Figura 60. Crear alarma a variable crítica

Para iniciar, se selecciona la casilla marcada en rojo en la Figura 60 y aparecerá la información de la Figura 61.

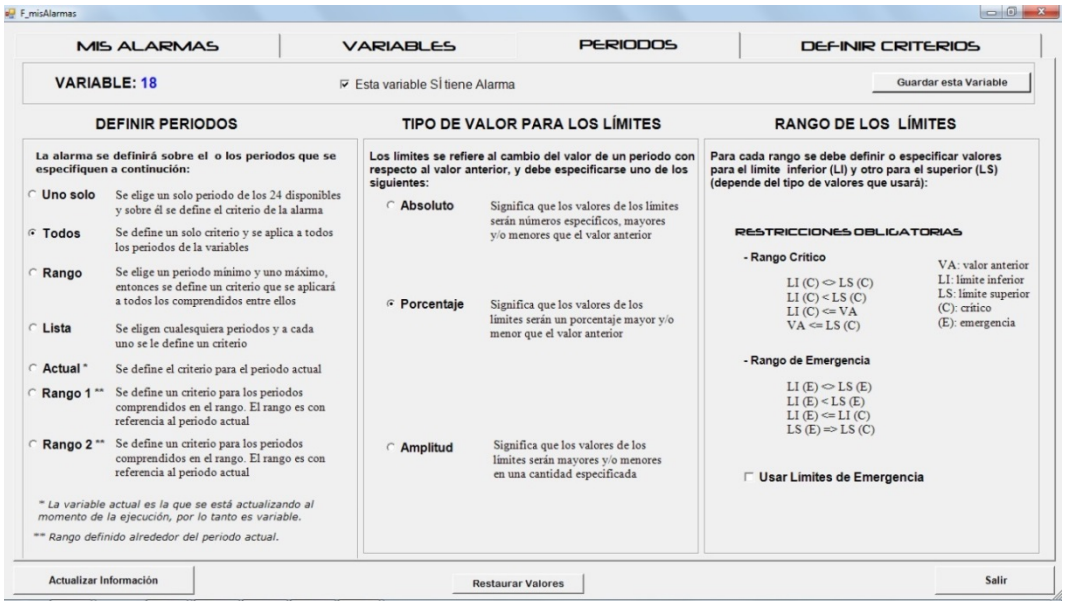


Figura 61. Definir modalidades de periodo y límites a variable crítica

Esta forma muestra los tipos de periodos y tipos de valores para los límites, también un esquema de los rangos que se pueden usar en el criterio. Hay 7 tipos de periodos a elegir y tres tipos de límites. Siempre que se elija alguno de estos en esta pestaña, se reseteará la información del criterio, por eso se debe tener cuidado cuando ya haya información de criterios de variables porque al cambiar de modalidad, ésta se perderá.

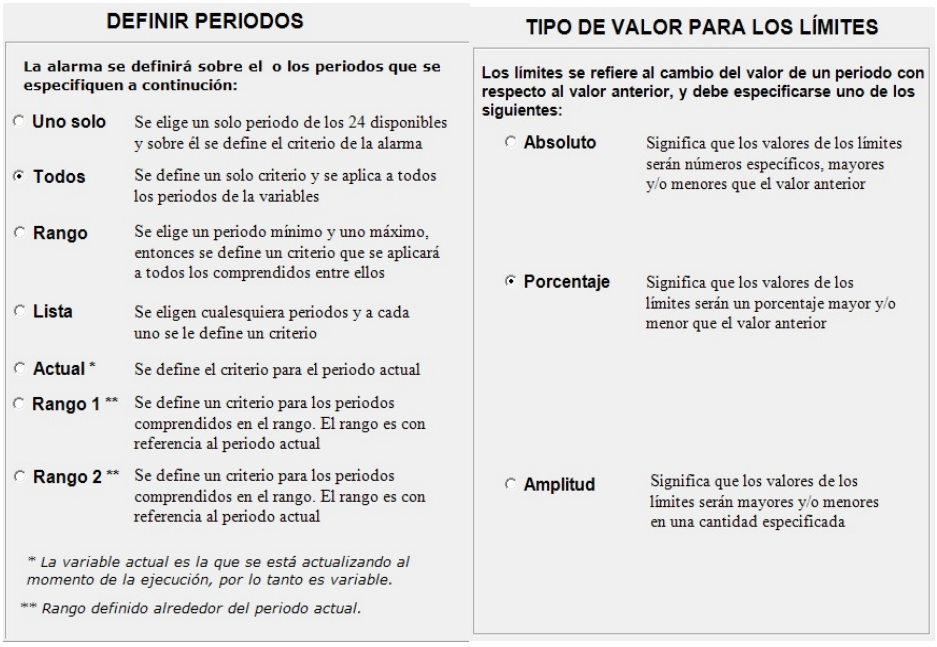


Figura 62. Definir periodos y límites

Figura 63. Elegir usar o no límites de emergencia

Si el cliente desea que su modelo no sea tan complicado, deja la casilla mostrada en la Figura 63 sin marcar y sólo tendrá que asignar un tipo de límite, el crítico. De ese modo cuando quiera usar estos límites debe marcarlo, y en el criterio ya estará disponible la posibilidad de poner límites de emergencia a cualquier variable.

Entendido lo anterior, se elige la pestaña “Definir criterios” y aparecerá la siguiente interfaz.

Figura 64. Definir criterios: selección de periodos

Antes de asignar un criterio a una variable, es necesario saber a cuántos y cuáles periodos se quiere asignar un criterio propio. En la barra de la imagen están numerados los 24 periodos de una variable cualquiera. En este caso como el tipo de periodo es “todos” es por lo que toda la barra es de color rojo, es decir, todos están seleccionados y el criterio que se decida crear va a afectar a los 24 periodos. En los demás casos de modalidad de periodo se puede seleccionar los periodos o el rango para el cual o los cuales se les asignará un criterio. Para iniciar con el criterio se selecciona la casilla “Asignar límites ahora” (ver Figura 64) y se muestran los límites de esta variable.

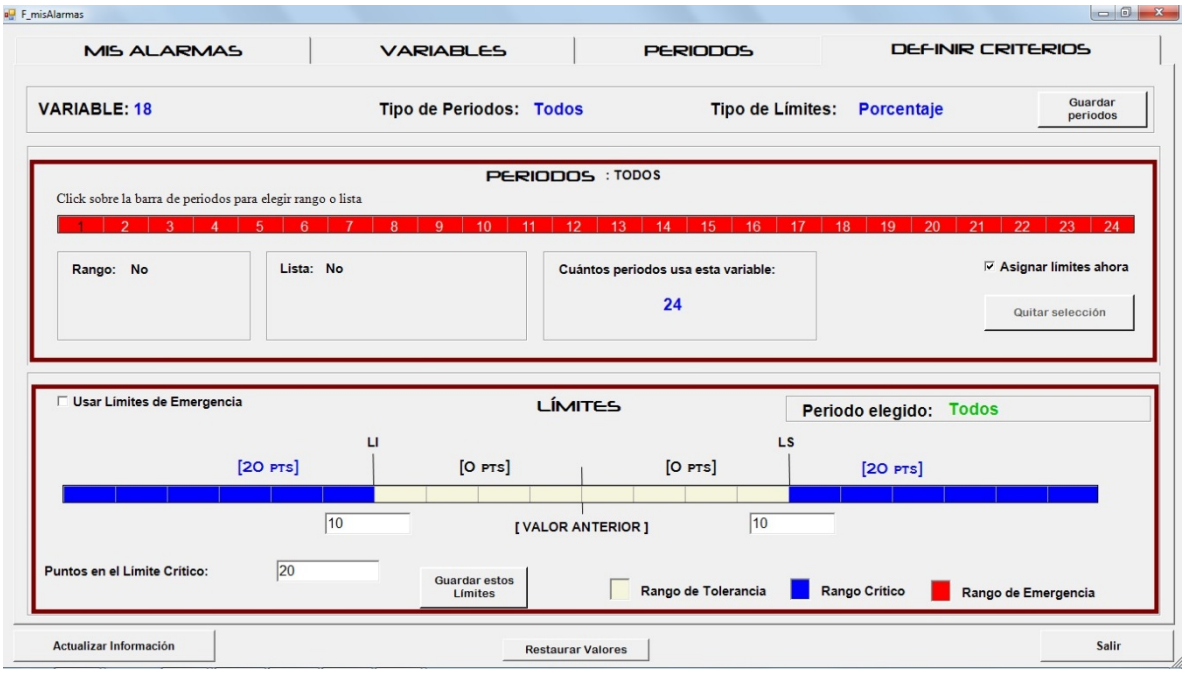


Figura 65. Definir criterios: límites y periodos

En la Figura 65 se puede ver en color azul de la barra el rango del límite crítico, también los puntos asignados al momento en el cuadro de texto de abajo a la izquierda, en este ejemplo es 20 puntos. También se pueden ver los cuadros de texto donde se debe capturar el valor de los límites inferior y superior críticos, en el ejemplo valen 10 y 10. Como se explicó antes el valor de estos límites es relativo al tipo de límites que seleccionó anteriormente, es decir, el número 10 puede ser 10%, 10 mayor al valor anterior o el número 10 en sí.

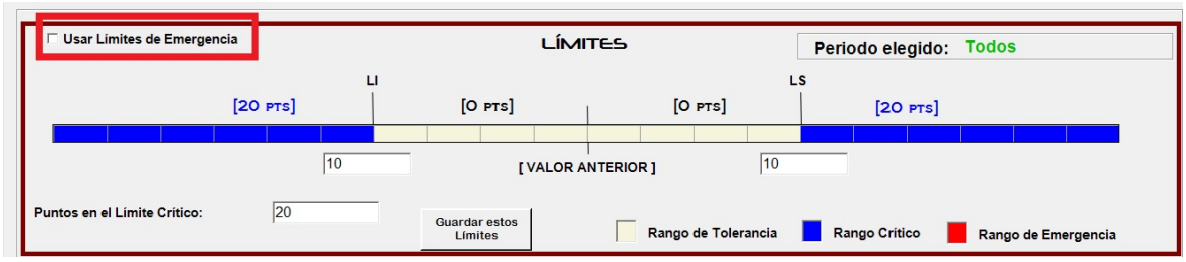


Figura 66. Límites: usar límites de emergencia

Si quiere usar límites de emergencia basta con marcar la casilla indicada en la Figura 66.

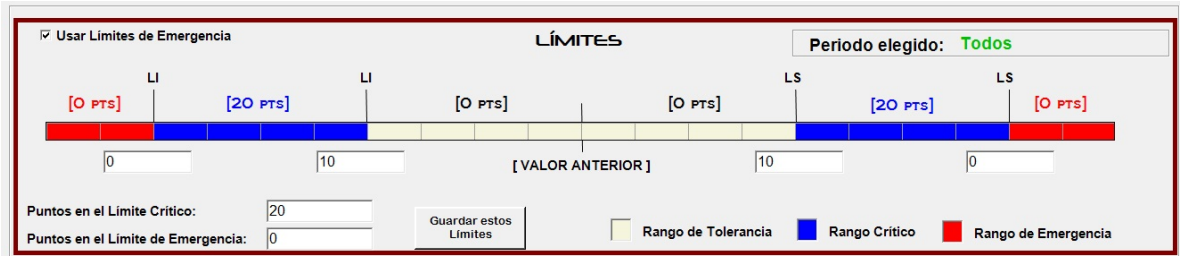


Figura 67. Criterio de alarmas con límites de emergencia

Ahora en la Figura 67 se puede ver en color rojo la zona de los límites de emergencia y los puntos asignados, por ahora cero. El cliente puede o no usar límites de emergencia según los necesite; es una buena herramienta para cuando quiere disparar avisos en casos muy específicos.

El programa de alarmas lee y edita las dos tablas de la base de datos mencionadas previamente cada vez que el cliente modifica sus criterios. Esta información se usa cuando una ejecución resulte de la aplicación de los criterios de ejecución, disparada por algún cambio de variables del mundo real o por decisión propia del cliente. Sin embargo la forma de usar estos datos se explica más adelante en la sección de preparar el archivo de fórmulas y en la ejecución.

3.7. Consultas al modelo

Quizá la función más útil, desde el punto de vista del cliente, es la consulta a su modelo para ver los valores de sus variables calculadas, puesto que ahí concluye el proceso de “informar”: recibe la información deseada.

La implementación de las consultas se diseñaron para proporcionar flexibilidad: el cliente puede ver el comportamiento de su modelo en ese momento. Como se explica en la siguiente sección, podrá “catalogar” algunas consultas que necesite ver constantemente. Esto le evitará tener que proporcionar los parámetros de la consulta cada vez que la necesite. Cuando invoque una tal consulta catalogada, podrá cambiar criterios y especificaciones, tanto para el uso presente o para modificarlos en la base de datos.

3.7.1. Tipos de consultas

Todas las variables que el cliente ha incluido en su modelo se pueden consultar. Entre éstas se incluyen las que él mismo ha definido, así como las del mundo real que haya incluido.

Las consultas se dividen en dos tipos principales de acuerdo a la forma de seleccionar las variables que se desean ver: mediante una lista de variables y por medio de una regla que permite armar un subconjunto de las variables. Técnicamente los dos tipos son listas, pero cuando se trata de una regla, las

variables son de un determinado tipo, es decir, se elige una determinada característica y la consulta despliega todas las que la posean.

Generalmente, el cliente tiene identificadas una serie de variables importantes para él, variables de las que quiere y debe estar informado a tiempo. Para esto puede definir consultas tipo lista con las variables que le interesan.

Las consultas tipo regla también son útiles, por ejemplo en los siguientes casos:

- Variables aleatorias
- Variables escalares
- Escalares cuyo número es mayor a 100 mil
- Variables del mundo real.

3.7.2. Datos que almacena una consulta (Tablas)

Como parte de los archivos del sistema, se incluye una base de datos denominada "queryCatalogue" con las consultas que se guardarán. Esta base tiene tres tablas denominadas "Consultas", "Consulta_lista" y "Consulta_regla" respectivamente.

Tabla 18. Tabla "consultas"

CAMPO	TIPO	DESCRIPCIÓN
Número de consulta	Entero	Consecutivo de consulta
Lista/Regla	Texto	L: lista; R: regla
Cómo mostrar	Byte	1: todas; 2: uno por uno
Rango/Lista/Periodos	Byte	1: todos; 2: rango; 3: periodos específicos
Periodo inicial	Byte	Periodo inicial de la consulta
Periodo final	Byte	Periodo final de la consulta
Cuántos periodos	Byte	Cantidad de periodos elegidos por el rango o la lista
Lista de periodos	Texto	Arreglo de periodos en esta consulta
Descripción	Texto	Comentarios (opcional)

La Tabla 18 muestra los atributos de una consulta que se almacena en la base de datos. Aquí se puede ver que cada consulta es única, debe ser lista o regla. También se debe indicar si se quieren mostrar todos los periodos o uno por uno. El tipo de selección de periodos puede ser: todos los períodos, un rango o por una lista de ciertos períodos específicos.

Se indican los periodos inicial y final. La cantidad de periodos usados se determina automáticamente al generar la consulta. La lista de periodos se refiere a los periodos usados, que pueden aparecer de forma consecutiva o salteada, según lo haya definido el cliente.

Tabla 19. Tabla "Consulta – Lista"

CAMPO	TIPO	DESCRIPCIÓN
Número de consulta	Entero	Consecutivo de consulta
Consecutivo en lista	Entero	Consecutivo dentro de la lista para esta consulta
Número de variable	Entero	

Tipo	Byte	VCLI, VMR, CTE
Tipo de valores	Byte	Escalar, aleatoria
Nivel	Entero	
Submodelo	Texto	
Unidades	Texto	
Shortie	Texto	
Descripción	Texto	

Como se dijo antes, una consulta consta de una lista de variables. En la Tabla 19 se ven los atributos para un registro de la lista de una consulta. En éste aparece la variable con todas sus características.

Tabla 20. Tabla “Consulta – Regla”

CAMPO	TIPO	DESCRIPCIÓN
Número de consulta	Entero	Consecutivo de consulta
Consecutivo de regla	Entero	Consecutivo dentro de las reglas para esta consulta
Modalidad	Byte	Se refiere al tipo de dato que se buscará
Valor a buscar	Texto	Valor que se buscará
Tipo de búsqueda	Byte	1: valor exacto; 2: contiene; 3: mayor; 4: menor
Tipo de variable a mostrar	Byte	0: todas; 1: VCLI; 2: VMR
Tipo de valor a mostrar	Byte	0: todas; 1: numérica; 2: distribuciones

Para el caso de que en una consulta haya varios tipos de reglas, en cada una se almacena la información mostrada en la Tabla 20. La característica principal de una regla es la modalidad, las cuales pueden ser de ocho tipos:

1. *Variable*. Específicamente una variable
2. *Submodelo*. Cuando se quiere buscar un submodelo, en el campo “valor a buscar” se especifica una variable o bien el submodelo de la misma. De esta forma se buscan las variables que son afectadas por este submodelo.
3. *Dad*. Se buscan las variables cuyo dad sea el indicado en el campo “valor a buscar”.
4. *Unidades*. Se buscan variables con las unidades indicadas.
5. *Nivel*. Variables con el nivel igual, menor o mayor al indicado aquí.
6. *Top variables*. Sólo las variables de arriba.
7. *Shortie*. Variables con nombre igual o parecido al indicado.
8. *Descripción*. Variables que contengan una descripción parecida a la indicada aquí.

Una vez decidida la modalidad e indicado el parámetro de búsqueda, se puede indicar el tipo de búsqueda, que depende de los dos anteriores. Por ejemplo, el “valor exacto” y “contiene” son aplicables cuando la modalidad es un texto ya que debe decidir si encontrar las palabras exactas o solo un extracto de ellas.

Las opciones “mayor” y “menor” se aplican cuando la modalidad es un número, es decir, cuando se quiere buscar variables mayores o menores a la que se indica o las que tiene nivel mayor o menor a ella.

Por último están los campos del “tipo de variable” y “tipo de valor” a mostrar, en los que se puede decidir si se quiere que la consulta muestre solo variables escalares o sólo variables aleatorias, variables del cliente o del mundo real o cualquier combinación entre ellas.

3.7.3. Especificación de una consulta

Las consultas inician con la interfaz mostrada en la Figura 68. Como se ve, están las características de cada consulta mostradas en la Tabla 18. En el ejemplo ya se han creado consultas previamente, sin embargo, ahora se presentará el ejemplo de cómo se crea una nueva.

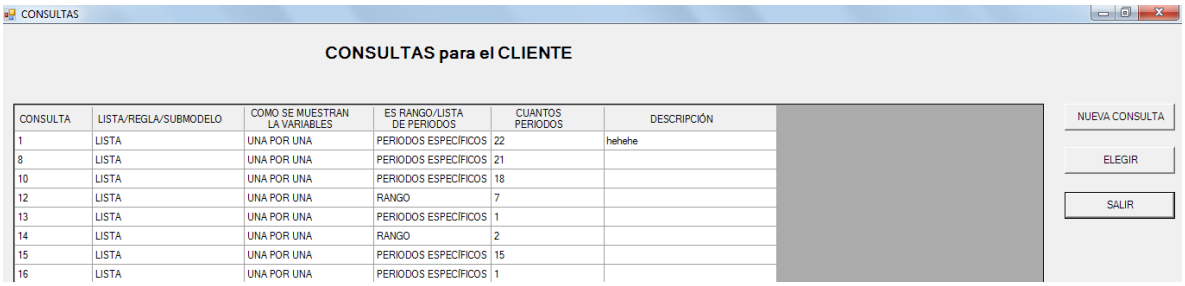


Figura 68. Interfaz de las consultas en FLAG

Para iniciar una nueva consulta se selecciona “nueva consulta” en el botón de la derecha, con lo que aparecerá la Figura 69. En ésta se decide el tipo de consulta que se va a crear: lista o regla. Para el ejemplo se crea una lista y selecciona el botón “siguiente”.

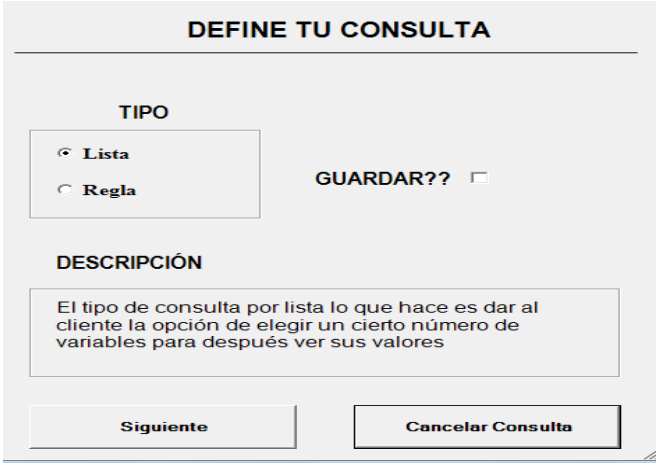


Figura 69. Definir nueva consulta

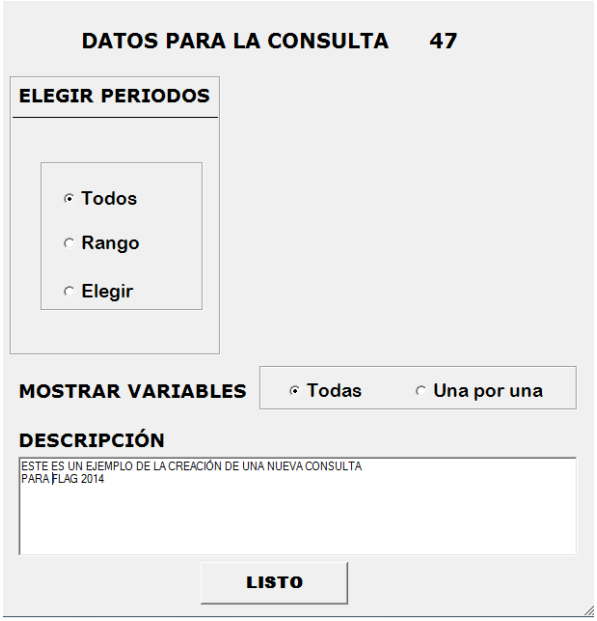


Figura 70. Características de una consulta

Después de elegir el tipo “lista” en la consulta, el cliente debe determinar el tipo de selección de periodos indicar cuáles quiere usar. En la Figura 70 se decidió que los periodos usados serían todos, es decir, los 24 de las variables. Y cuando se muestren, será una por una cada variable de la consulta.

En cambio, en la Figura 71 se muestra un ejemplo de la selección de los periodos como un rango. El inicial es el número 3 y el final el 15, por lo que la consulta mostrará en total 13 periodos.

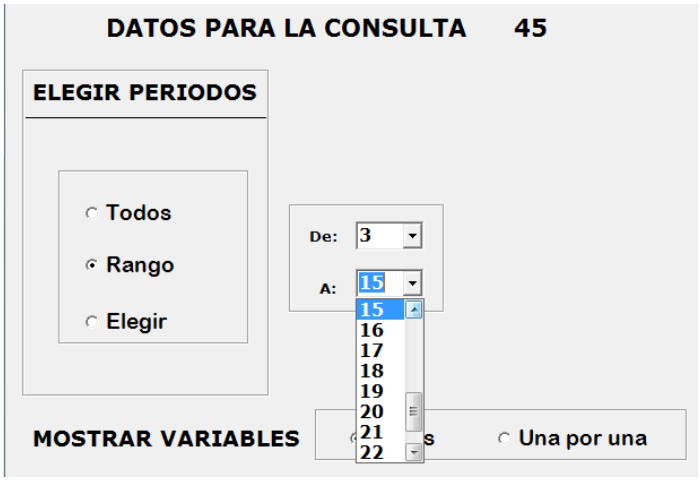
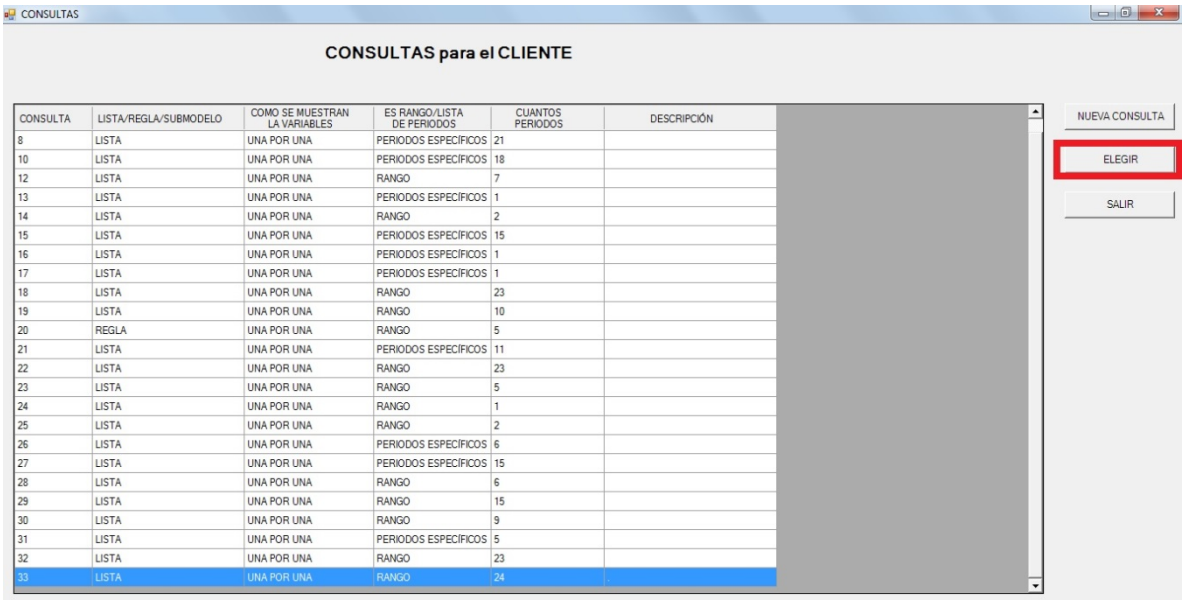


Figura 71. Consulta tipo “rango de periodos”

Figura 72. Consulta tipo “periodos específicos”

Si desea indicar “períodos específicos”, se muestra una forma como la que ilustra la Figura 72 donde se escogen los que el cliente desee ver en su consulta.

Una vez hecho esto, el programa regresa a la interfaz principal donde ya aparece la nueva consulta que se ha creado (ver Figura 73).



CONSULTA	LISTA/REGLA/SUBMODELO	COMO SE MUESTRAN LA VARIABLES	ES RANGO/LISTA DE PERIODOS	CUANTOS PERIODOS	DESCRIPCIÓN
8	LISTA	UNA POR UNA	PERIODOS ESPECÍFICOS	21	
10	LISTA	UNA POR UNA	PERIODOS ESPECÍFICOS	18	
12	LISTA	UNA POR UNA	RANGO	7	
13	LISTA	UNA POR UNA	PERIODOS ESPECÍFICOS	1	
14	LISTA	UNA POR UNA	RANGO	2	
15	LISTA	UNA POR UNA	PERIODOS ESPECÍFICOS	15	
16	LISTA	UNA POR UNA	PERIODOS ESPECÍFICOS	1	
17	LISTA	UNA POR UNA	PERIODOS ESPECÍFICOS	1	
18	LISTA	UNA POR UNA	RANGO	23	
19	LISTA	UNA POR UNA	RANGO	10	
20	REGLA	UNA POR UNA	RANGO	5	
21	LISTA	UNA POR UNA	PERIODOS ESPECÍFICOS	11	
22	LISTA	UNA POR UNA	RANGO	23	
23	LISTA	UNA POR UNA	RANGO	5	
24	LISTA	UNA POR UNA	RANGO	1	
25	LISTA	UNA POR UNA	RANGO	2	
26	LISTA	UNA POR UNA	PERIODOS ESPECÍFICOS	6	
27	LISTA	UNA POR UNA	PERIODOS ESPECÍFICOS	15	
28	LISTA	UNA POR UNA	RANGO	6	
29	LISTA	UNA POR UNA	RANGO	15	
30	LISTA	UNA POR UNA	RANGO	9	
31	LISTA	UNA POR UNA	PERIODOS ESPECÍFICOS	5	
32	LISTA	UNA POR UNA	RANGO	23	
33	LISTA	UNA POR UNA	RANGO	24	

Figura 73. Nueva consulta agregada. Seleccionarla y “elegir” para editarle

Para agregar variables a una consulta, se la selecciona y se usa el botón “ELEGIR”, lo cual nos dirige a la Figura 74. Para este caso, la consulta seleccionada es la número 33, la cual no cuenta con variables para mostrar.

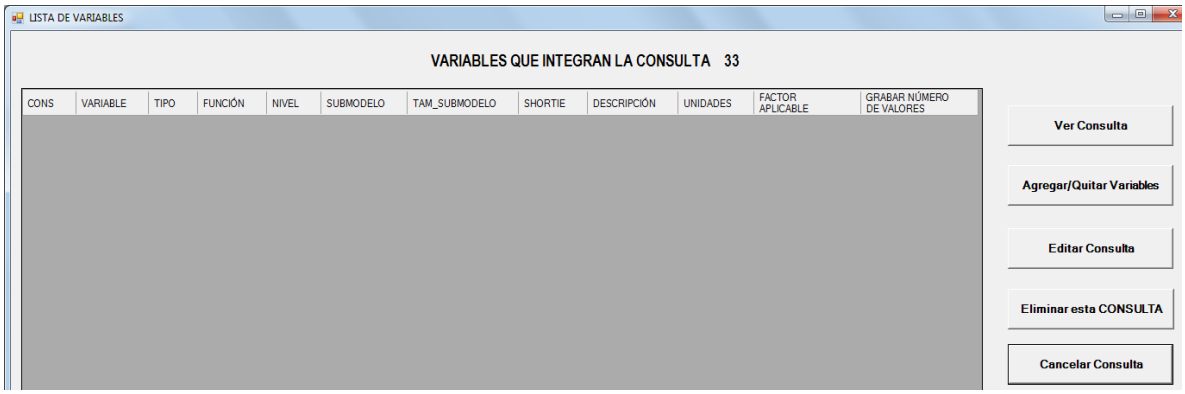


Figura 74. ABC de variables en una consulta tipo lista

En la parte derecha se pueden observar cinco botones, de los cuales el primero es el que permite ver la consulta, es decir, los valores de las variables mostradas. Por ahora, como no hay variables agregadas, no se pueden consultar valores. Con el segundo botón aparece la Figura 75, en la que se pueden agregar y quitar variables del cliente y variables del mundo real. Cuando se han agregado variables a la consulta, como lo muestra la Figura 76, éstas aparecen en la interfaz del tipo de consulta, que en un principio estaba vacío (ver Figura 77).

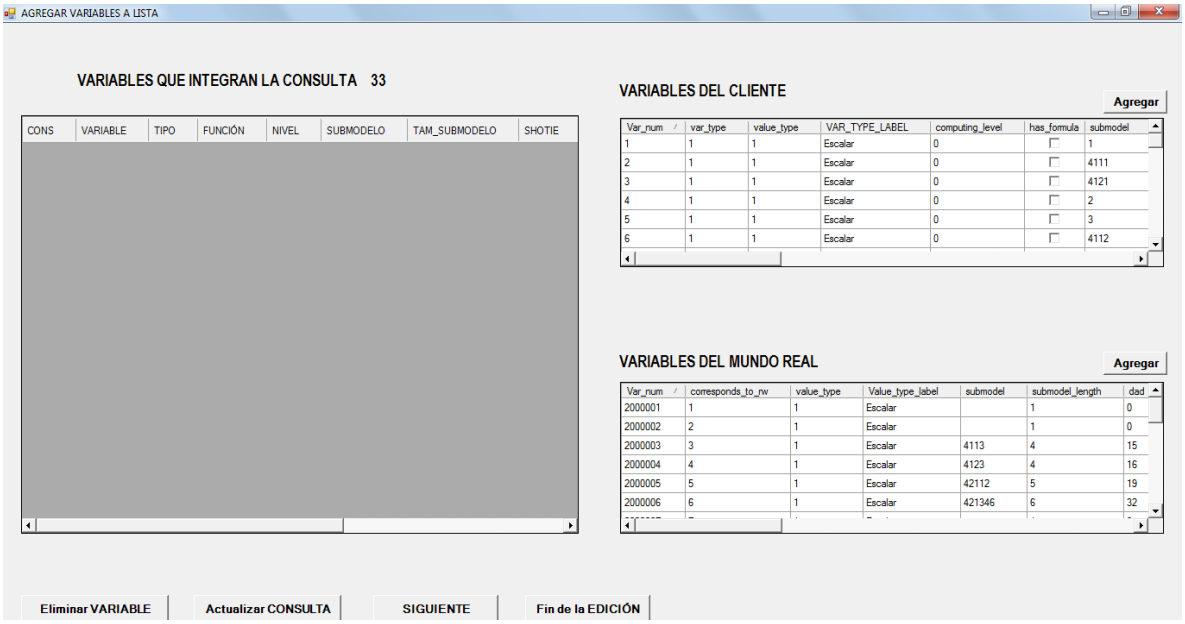


Figura 75. Agregar/Quitar variables a una consulta

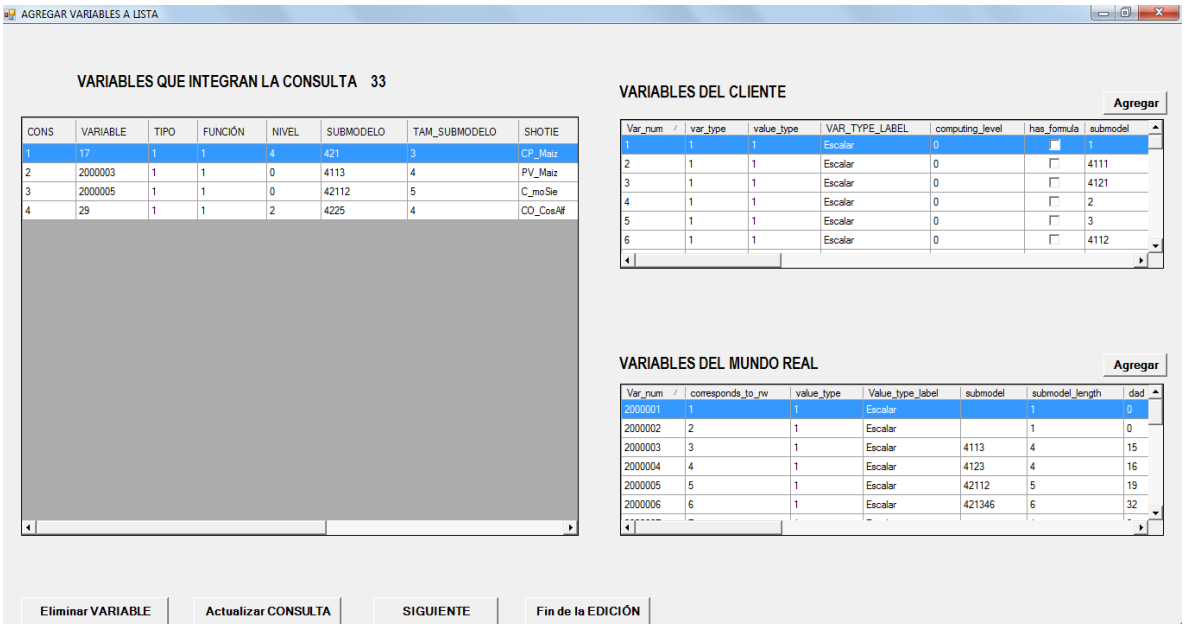


Figura 76. Cuatro variables agregadas a la consulta

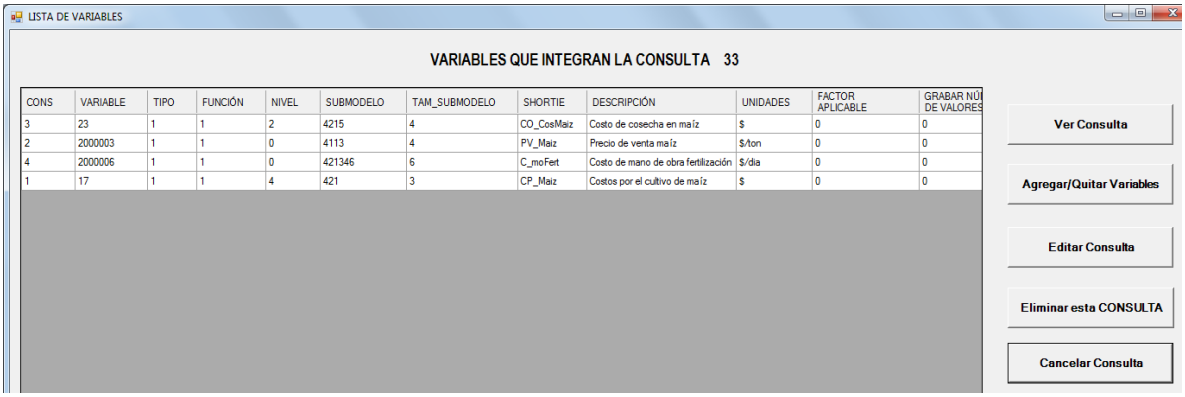


Figura 77. Las mismas cuatro variables en el esquema de edición de la lista de la consulta

El botón “editar consulta” manda a la interfaz de la Figura 70, donde el cliente puede hacer algunos cambios a su consulta. Los dos siguientes botones eliminan la consulta y cancelan la edición respectivamente. Ahora como ya hay variables, se puede “ver la consulta”.

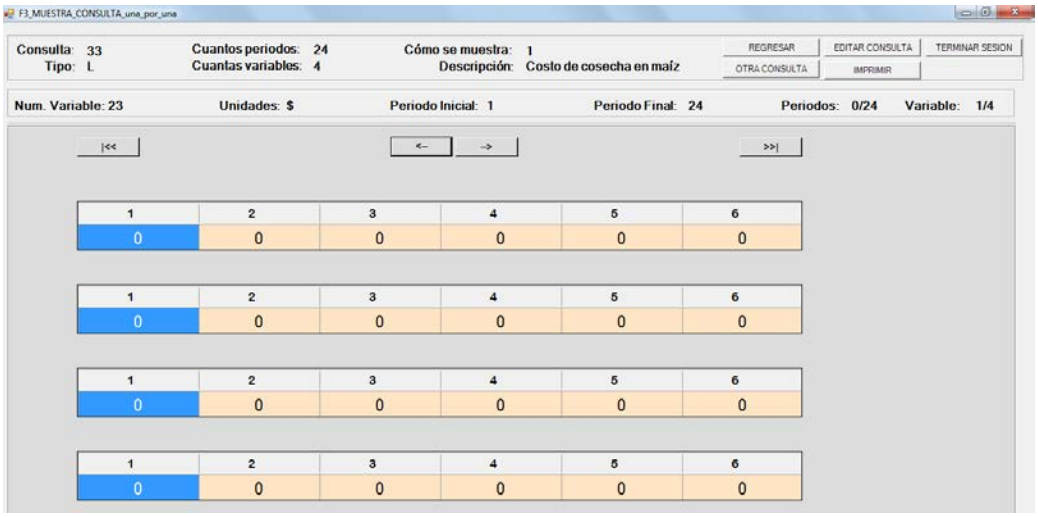


Figura 78. Consulta número 33, variable número 23

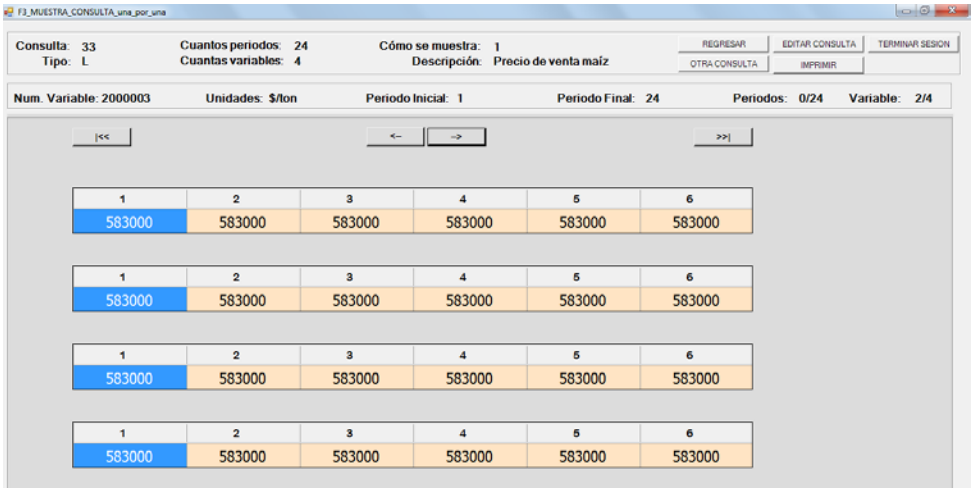


Figura 79. Consulta número 33, variable número 2000003

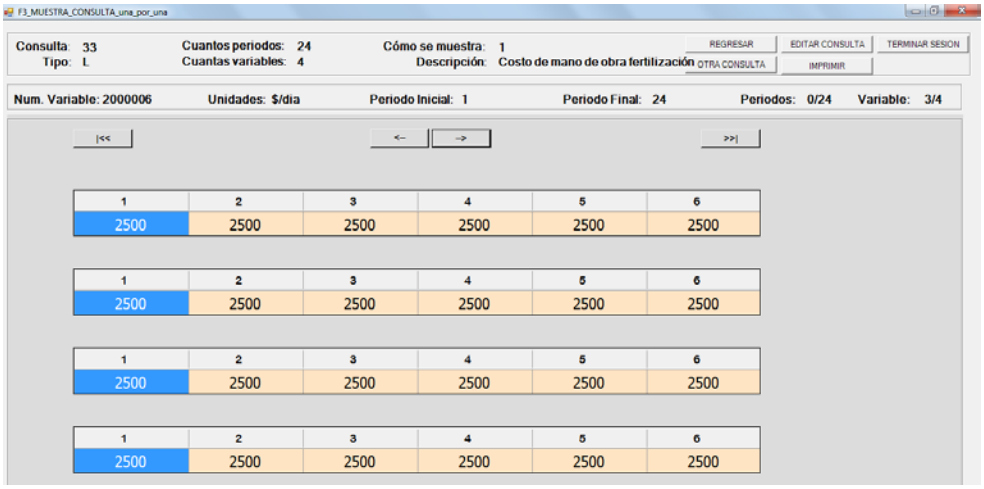


Figura 80. Consulta número 33, variable número 2000006

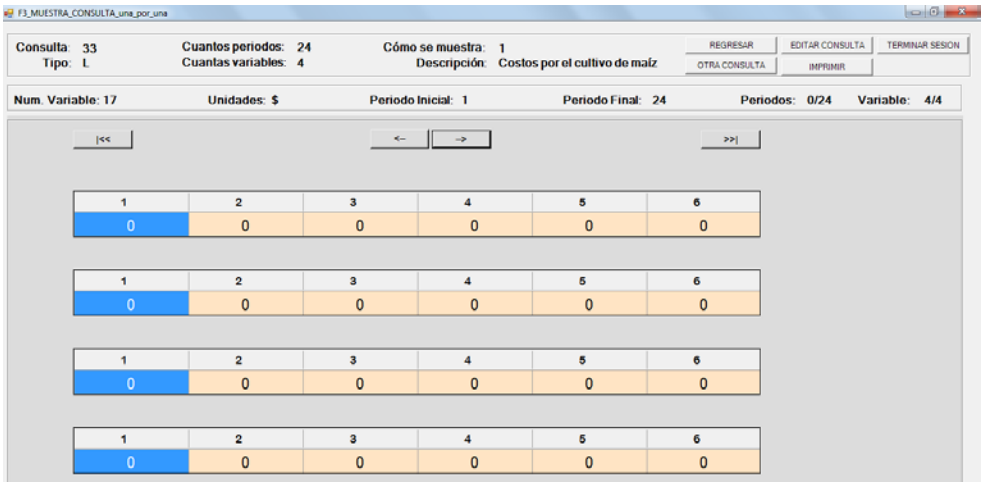


Figura 81. Consulta número 33, variable número 17

La interfaz que muestra los valores se observa en las figuras de la Figura 78 a la Figura 81; en cada una la variable correspondiente con sus valores de los 24 periodos (para este caso). En la parte de arriba están los datos de la consulta: número, periodos usados, cómo se muestra (el 1 indica que es una por una las variables), el número de variables de esta consulta y el tipo (para este caso es una lista).

3.8. Gráfica del modelo

3.8.1. Introducción

Se agregó a esta nueva versión de FLAG un módulo que se encarga de crear un esquema del modelo que ha introducido un cliente. Dicho esquema se realiza mediante un objeto incluido en Visual Studio 2010 denominado *tree view*.

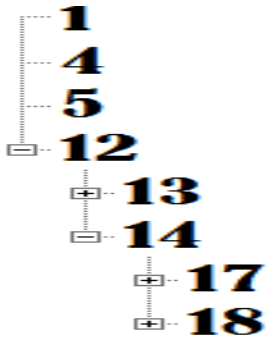


Figura 82. Ejemplo del objeto *tree view* en .NET

Como el nombre lo indica, se trata de una “vista de árbol” – por su traducción al castellano – y se adapta perfectamente para representar los modelos que se crean en FLAG que son precisamente de tipo árbol. Antes se explicó que los modelos de

FLAG constan de fórmulas y variables que constituyen los operandos de las mismas. En la Figura 82 se puede mostrar esto, por ejemplo, tomando al número 12 como una fórmula: dentro de ésta aparecen dos operandos, las variables 13 y 14. A su vez, la variable 14 contiene dos operandos y así sucesivamente.

Se usa dicho objeto para mostrar el modelo de un cliente. El proceso que se encarga de generar el dibujo se encuentra en un módulo aparte, separado del sistema principal, es decir, en una dll, donde están las rutinas y algoritmos usados, así como la interfaz que lo muestra. En seguida se dará una breve explicación del módulo que dibuja el modelo.

3.8.2. Interfaz inicial

El módulo que crea el dibujo siempre lo hace con el modelo más actual, es decir, el último que no ha recibido cambios. Para esto es necesario leer la base de datos y verificar que el programa de "ejecución de niveles y submodelos" haya sido ejecutado, de lo contrario el actual módulo no permite generar el dibujo. Lo anterior es debido a que se deben leer los submodelos de la base de datos, o para que se entienda mejor, los mostrados en la Figura 92.

Al inicio del programa, se extrae la información mostrada en la Tabla 21 y se manda a memoria para su uso posterior.

Tabla 21. Información necesaria para dibujar el modelo

CAMPO	TIPO	DESCRIPCIÓN
Número de variable	Entero	
Shortie	Texto	Nombre corto de la variable
Tipo	Byte	Puede ser VCLI, VMR o algún sinónimo
Corresponde a	Entero	Se refiere al número de la VMR en FLAG o a la variable que reemplazó un sinónimo
Submodelo	Texto	Numeración en formato texto que determina la posición de la actual variable dentro del modelo.

Como se podrá intuir, el campo más importante es el submodelo ya que es el que determina dónde se encuentra una variable dentro del modelo generado por un cliente. Para entender mejor esta parte conviene leer la sección 4.1.9 Generación de submodelos.

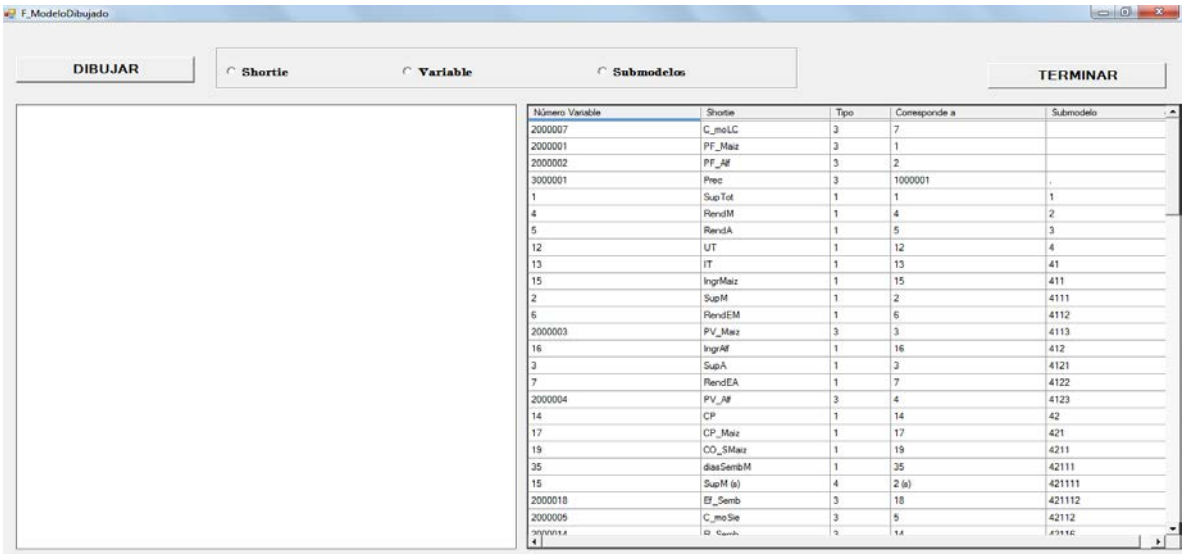


Figura 83. Interfaz principal del módulo que dibuja el modelo

Se crea el arreglo de todas las variables del modelo (VCL, VMR y sinónimos).

Para ello se procesan las tres tablas correspondientes de la base de datos del cliente. A continuación se orden el arreglo por submodelo (observe que este dato en la tabla es de tipo carácter (string) de modo que se hace un sort alfabético.

Una vez creado este arreglo de las variables que hay en el modelo con la estructura mostrada en la Tabla 21, se prosigue a mostrar la interfaz principal en la Figura 83. Se puede observar una tabla de datos en la parte derecha con los datos que fueron extraídos de la base. En la parte de arriba hay un cuadro de opciones de las que el cliente podrá elegir cómo visualizar su dibujo, esto es, si quiere ver el nombre de la variable, el número o el submodelo. También aparecen dos botones, uno para crear y mostrar el dibujo y el otro para terminar.

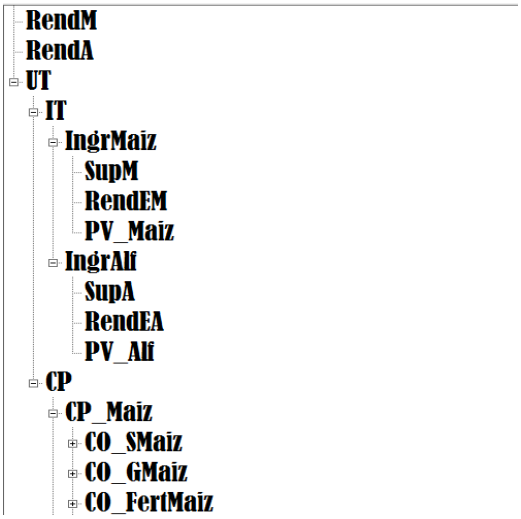


Figura 84. Modelo dibujado con la opción “shortie”

Cuando se selecciona la opción “shortie” (el nombre corto de la variable), el modelo dibujado se ve de la forma mostrada en la Figura 84. Se puede ver que la fórmula IT tiene dos operandos, IngrMaiz e IngAlf. Por su parte IngrMaiz tiene tres operandos: SupM, RendEM y PV_Maíz. Estos últimos no son fórmulas por lo que no contienen operandos. Otra manera de ver lo anterior es como se ve en la Figura 85.

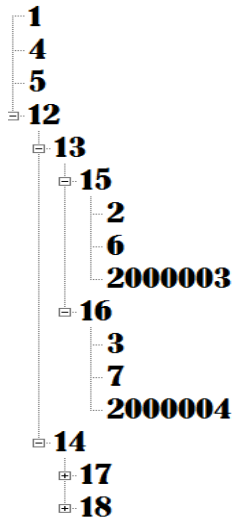


Figura 85. Modelo dibujado con la opción “variable”

Ahora en lugar de ver el nombre de cada variable, aparecen los números de cada una. Y para el submodelo sería como lo muestra la Figura 86.

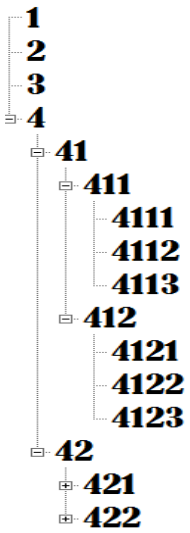


Figura 86. Modelo dibujado con la opción “submodelo”

En esta parte se puede notar que hay tres submodelos separados del principal, que es el número 4. Para este ejemplo la variable de arriba es la número 12 y corresponde al submodelo 4, por lo que todas las demás fórmulas están por

debajo de ella, contenidas en sus operandos. Pero qué pasa con los submodelos 1, 2 y 3, es decir, las variables 1, 4 y 5; éstas son variables de arriba pero no son operandos en ninguna otra, esa es la razón por la que aparecen como submodelos independientes.

3.8.3. Algoritmo que genera el árbol

El objeto tree view está compuesto por una serie de nodos, como se puede apreciar en las figuras ya mostradas. A tales nodos es posibles crearles subnodos y a éstos otros subnodos, y así sucesivamente. Usando esta característica del objeto es como se explicará el algoritmo usado para dibujar el modelo.

Primero se lee el arreglo – que ya está en memoria - de las variables y sus características. Para cada una se ejecuta una rutina recursiva de la siguiente forma:

```
While p < misVariables.length
        If misVariables(p).submodelo <> “” _
                 and misVariables(p).submodelo <> “0” then _
                dibujaSubmodelo(misVariables(p).submodelo, 1)
        p += 1
End while
```

donde:

p: variable que funge como contador
misVariables: arreglo que contiene a las variables y su información

La rutina recursiva denominada “dibuja submodelo” recibe como parámetro el submodelo de la variable en proceso y un número 1. Éste último es porque es la primera vez que se procesa dicha variable. Más adelante se verá cómo va cambiando este parámetro. La rutina recursiva tiene las siguientes instrucciones:

```
Public Sub dibujaSubmodelo(ByVal submodelo As String, ByVal x As Int32)
    Dim nodo As String

    nodo = (Left(submodelo, x))
    If existeNodo(nodo) Then
       dibujaSubmodelo(submodelo, x + 1)
    Else : creaNodo(submodelo)
    End If
End Sub
```

Aquí se determina el nodo del submodelo que llega, para lo cual se usa el número contenido en “x”. En el primer caso es un 1, por lo que el nodo, según nuestro ejemplo, será el 4. Después se pregunta si existe este nodo en el objeto tree view, de ser así se procede a “dibujar el submodelo” hijo del que llegó. De hecho se manda el mismo que llegó, pero ahora se suma 1 al parámetro “x”, por lo que el nodo siguiente que se obtendrá será uno de tamaño 2.

Para determinar si existe el nodo, se usa la siguiente rutina:

```
Public Function existeNodo(ByVal nodo As String) As Boolean
    existeNodo = False
    With F_ModeloDibujado.TV_modelo
      If .Nodes.Find(nodo, True).Length = 1 Then
        nodoDAD = .Nodes.Find(nodo, True)(0)
        Return True
      End If
    End With
End Function
```

El nodoDAD que aparece en la rutina es una variable tipo “treeNode” declarado previamente. Se usa para almacenar el nodo padre que invocó dicha rutina, pero solo en el caso de encontrar el nodo dentro de la estructura tree view.

Cuando el nodo no existe, se prosigue a créalo, para lo cual se ejecuta primero la siguiente rutina:

```
Public Sub creaNodo(ByVal submodelo As String)
    With F_ModeloDibujado.TV_modelo
      If nodoDAD Is Nothing Then
        ejecuta_dibuja(submodelo, 1)
      Else
        ejecuta_dibuja(submodelo, 2)
        nodoDAD = Nothing
      End If
    End With
End Sub
```

Aquí se hace notar que cuando ya existe un dad, el segundo parámetro de la rutina “ejecuta dibuja” es 2, de lo contrario es 1. En seguida se detalla esto.

```
Public Sub ejecuta_dibuja(ByVal submodelo As String, ByVal tipo As Byte)
      With F_ModeloDibujado.TV_modelo

        If tipo = 1 Then
          Select Case mostrar
            Case "submodelo"
              .Nodes.Add(submodelo, misVariables(p).submodelo)
          End Select
        Else
          Select Case mostrar
            Case "submodelo"
              nodoDAD.Nodes.Add(submodelo, misVariables(p).submodelo)
          End Select
        End If
      End With
End Sub
```

El parámetro tipo se refiere a si el nuevo nodo que se ha de crear será sobre la raíz del árbol o será sobre otro nodo que ya fue creado. En caso de ser este último, se usa la referencia al susodicho nodo en la variable nodoDAD que antes fue asignada cuando se determinó que sí existe el dad del nodo en proceso.

Para simplificar el ejemplo se omitieron las sentencias que asignan los otros tipos de etiqueta a mostrar, es decir, el número de la variable o el shortie. Éstas serían otros “casos” dentro del contenedor *select*.

Resumiendo, se crean en primer lugar los nodos pertenecientes a los submodelos 1, 2, 3 y 4. En seguida se crean el 41, 42, …, etc. Después el 411, 412, …, etc. Para crear el nodo 41, por ejemplo, el nodo padre de éste (el 4) ya existe, por lo que el nodo 41 se genera como hijo del número 4.

Cuando han sido creados todos los nodos necesarios, termina la rutina y procede a mostrar el tree view generado.

4. SISTEMA FLAG ADMINISTRADOR

4.1. Preparación del Archivo de Fórmulas

En esta sección se explicará todo el proceso que se lleva a cabo para preparar el archivo de fórmulas, la parte central del modelo de cada cliente. Se trata de entregar las fórmulas al programa que las ejecuta de modo que minimice la duración del proceso de cálculo. Esto adquiere mayor importancia a medida que aumente el número de modelos que se recalcularán a partir de cambios en los valores de las variables del mundo real obtenidos por los agentes.

Para lograr mayor eficiencia, se determinaron varios aspectos de la preparación:

- Se calculan los llamados *niveles* de las fórmulas, que permiten que éstas se ejecuten en el orden debido. Una fórmula no se podrá recalcular si no se ha hecho lo propio con todos sus operandos. Durante este proceso, se detectan ciclos si los hubiera.
- Se prepara una lista – ordenada por nivel - de las fórmulas, para que se ejecuten en el orden indicado por sus respectivos niveles. Cabe señalar que la versión anterior no contaba con una lista como ésta, lo que resultaba en que se leían las fórmulas del archivo una y otra vez, ejecutando en cada una de las lecturas las fórmulas del nivel siguiente a los de la anterior.
- También se sustituyen operandos “repetidos” (que aparecen en más de una fórmula) por sinónimos, que indican que a pesar de tener una designación distinta, usan los mismos valores que el operando que no se ha reemplazado por un sinónimo. En la versión anterior del FLAG, el cliente creaba estos sinónimos; ahora se transparentó este concepto de modo que el cliente no lo conoce. Cabe señalar que la necesidad de incluir sinónimos tiene dos fines: permite la construcción de los submodelos, y hace posible la diagramación del modelo para su revisión o permitir su entendimiento en forma fácil.
- Finalmente, se preparan los submodelos que se ejecutan cuando sólo ha cambiado una sola variable del mundo real. Para esto, se determinan aquellos operandos que son aditivos en las fórmulas en las que intervienen.
- Adicionalmente, se decidió que el programa de ejecución de fórmulas no tuviera necesidad de utilizar tablas de las bases de datos de los clientes. Para ese fin, se agregaron al archivo de fórmulas dos elementos: los criterios de alarmas del cliente y sus datos de identificación, en especial, los que sirven para el envío de las alarmas generadas durante la ejecución.

Se describe en este capítulo el archivo de fórmulas resultante de la preparación con todos los elementos descritos, y cada uno de los procesos que resultan en la información contenida en dichos archivos.

4.1.1. Esquema General del Archivo de Fórmulas

El cliente introduce las fórmulas que necesita su modelo. Se almacenan en un archivo plano, en lugar de usar una tabla de su base de datos. Este proceso ya se comentó en la sección dedicada a dicho tema. De ese modo, cuando termina el proceso de definición de las fórmulas, hay un archivo que contiene precisamente eso: las fórmulas. Sin embargo, se aprovecha este archivo para incluir la información que necesita el sistema para generar alarmas de acuerdo a los criterios definidos para tal efecto por el cliente. Como se verá, se agregan también los elementos necesarios para la ejecución de submodelos en lugar de recalcular todas las fórmulas. Este concepto se explica más abajo.

El archivo de fórmulas que se entrega a la ejecución de los modelos (que se prepara a partir del archivo que construye el cliente) contiene toda la información mencionada en la sección anterior. Es importante señalar que se trata de un archivo que se usa como “binary”, es decir, se obtienen sus elementos a partir de su posición relativa al inicio del archivo. De ese modo, se introducen apuntadores que permiten determinar dónde se encuentran los elementos cuando surja la necesidad de usarlos. También es importante señalar que este archivo se crea a partir del que le llega de fórmulas (es decir, sustituye el archivo anterior).

Para ello, el archivo contiene diversas estructuras que se mencionan a continuación. Esto resulta en un archivo “complicado”, es decir, no es fácil entender cómo se guarda y usa la información. Para efectos de facilitar la lectura y la comprensión de estas estructuras, se han numerado las diversas “secciones” o partes del archivo en esta descripción (ver Tabla 22).

Tabla 22. Composición del archivo de fórmulas

Parte	Qué contiene	Observaciones
1	Registro base	Apuntadores a las otras partes
2	Lista ordenada de fórmulas	Apuntan a los respectivos registros
3	Las fórmulas mismas	1 registro por cada fórmula
4	Lista de submodelos	Cuáles submodelos (apuntan a la lista de fórmulas)
5	Lista de fórmulas por submodelo	
6	Información general de alarmas	También contiene los datos del cliente
7	Criterios de alarma	Cada registro contiene la información de cada alarma existente

Cada parte del archivo está compuesta por un tipo de estructura específico y a excepción de la parte 1 y 5, son arreglos. En la Tabla 23 se presentan los campos que componen las primeras tres partes del archivo; en la Tabla 24 están las partes

restantes del archivo. Cada una de ellas se explicará más adelante con todo detalle, informando el uso de cada campo, su importancia, la ubicación de la información usada y los procesos que generan los arreglos.

Tabla 23. Estructuras de cada parte del archivo de fórmulas (parte 1)

1	2	3
Registro base	**Lista ordenada de fórmulas**	**Una fórmula**
Posición de la primera fórmula	Variable	Número de variable
Número de fórmulas	Nivel	Nivel
Cuántos submodelos hay	Registro relativo	Factor aplicable
Dónde empieza lo de submodelos		Subtotal ()
Número de variables con alarma		Cuántos subtotales
Dónde empieza lo de alarmas		Cuántos sinónimos
Fecha último cálculo de niveles		
Fecha último cambio a fórmulas		

Tabla 24. Estructuras de cada parte del archivo de fórmulas (parte 2)

4	5	6	7
Lista de submodelos	**Listas de fórmulas**	**Información general de alarmas**	**Alarmas por variable**
Número de la variable	Número de variable	E – mail	Número
Número de la variable (para el cliente)	Posición de su fórmula	Teléfono	Nombre
Cantidad de fórmulas en el submodelo	Hija	Nombre	Shortie
Posición de la lista	Nivel	Variables con alarma	Tipo Límites
		Puntos de cada rango (4)	Tipo periodo
			Periodo cero
			Parámetros por periodo (24)

Las estructuras y las partes del archivo se diseñaron específicamente para mayor eficiencia, y se tomó en cuenta el grado de complejidad que pudieran representar en el que quisiera conocerlas.

La generación de este archivo consiste de los siguientes procesos:

- Cálculo de niveles
- Armado de la lista de fórmulas en orden de ejecución
- Generación de sinónimos
- Determinar operandos aditivos de fórmulas
- Generar submodelos
- Construir las listas de fórmulas que se ejecutan como parte de los submodelos

- Extraer criterios de alarmas de la base e incluirlos en el archivo
- Copiar datos del cliente al archivo (teléfono, email, nombre)
- Actualización de la base de datos (con los sinónimos definidos y los submodelos calculados).

Se diseñó este programa para incrementar su eficiencia computacional. Esto se hizo a pesar de que no implica una ejecución larga, dada la exigencia de que los programas y rutinas deberían ser lo más eficientes posibles, especialmente si forman parte de un proyecto de investigación. El mismo comentario aplica a muchas otras rutinas de este programa de preparación de fórmulas.

Para mayor eficiencia, se fijaron criterios:

- Minimizar las operaciones de input-output. En particular, no leer muchas veces "lo mismo".
- Cuando se trata de grabar un registro que luego se actualizará con datos complementarios, no se graba sino se arma en memoria y se graba cuando está listo.
- Si hay diversas rutinas que usan o actualizan los mismos campos, se intentará juntarlas para que lo hagan simultáneamente.
- Si se usan tablas de una base de datos, preferiblemente se arman una única vez los *datasets* correspondientes (esto puede requerir filtros o reordenamientos).

4.1.2. Registro base de fórmulas (parte 1 del archivo)

Este registro es el que permite el uso correcto de todas las partes del archivo. Cabe señalar que, aunque se "crea" este registro como primer paso del proceso de preparación, los restantes procesos lo actualizan (con los apuntadores respectivos) a medida que se agregan los restantes elementos del archivo.

Los campos del registro base se explican después de repetir, para conveniencia del lector, la estructura del registro (ver Tabla 25).

Tabla 25. Estructura del registro base del archivo de fórmulas

CAMPO	TIPO	DESCRIPCIÓN
Posición de la primera fórmula	Entero	Se refiere al número de byte en el que empiezan a guardarse las fórmulas capturadas. Esta posición depende de las estructuras iniciales que están primero, como lo es esta misma y la de variables ordenadas.
Número de fórmulas	Entero	Este es el número total neto de fórmulas que hay guardadas.
Cuántos submodelos hay	Entero	Es el número de listas de fórmulas para submodelos que hay almacenadas
Dónde empieza lo de submodelos	Entero	Es el número de byte en el archivo en el que comienza la información de los submodelos

Número de variables con alarma	Entero	Sólo las que tienen definida una alarma
Dónde empieza lo de alarmas	Entero	Número de byte en el archivo donde empieza la información de alarmas.
Fecha último cálculo de niveles	Fecha	Si esta fecha es anterior a la fecha del último cambio de fórmulas, se deben recalcular los niveles
Fecha último cambio a fórmulas	Fecha	Fecha de la última edición a fórmulas

Se puede observar que además de contener los apuntadores a las otras "partes" del archivo, también contiene información de las fechas en las cuales se lo actualizó. Esto permite evitar que se ejecuten procesos redundantes (es decir, volver a crear este archivo cuando no ha habido cambios a fórmulas o criterios de alarma del cliente).

4.1.3. Variables ordenadas (parte 2 del archivo)

En esta parte se almacena un arreglo de "fórmulas a ejecutar" ordenadas en forma ascendente por el nivel de la fórmula. Esto permite al programa de ejecución calcularlas sin violar la restricción: no se puede ejecutar una fórmula antes de hacer lo propio con todos sus operandos.

Tabla 26. Estructura de variables ordenadas

CAMPO	TIPO	DESCRIPCIÓN
Variable	Entero	Es el número de la variable a la que corresponde la fórmula en esta posición
Nivel	Entero	Es el nivel que le corresponde a esta fórmula en el árbol del modelo.
Registro relativo	Entero	Es la posición en el archivo en la cual está guardada la fórmula correspondiente a la variable

Cada variable con fórmula capturada tiene los atributos mostrados en la Tabla 26. Inicialmente el nivel es cero para las variables capturadas y su registro relativo es el consecutivo de su captura. Estos dos campos cambian al ejecutarse el cálculo de niveles y submodelos que se incluye en el proceso actual de preparación del archivo. Un arreglo de estructuras del tipo *variables ordenadas* es la información que se guarda inmediatamente después de la estructura *registro base*. El tamaño de la misma es el número de fórmulas en el modelo multiplicado por la longitud de cada una de sus fórmulas (10 bytes).

4.1.4. Estructura de una fórmula (parte 3 del archivo)

Cada fórmula, en el modelo de cliente, tiene la estructura mostrada en la Tabla 27. Dentro de ella se encuentran datos importantes para la ejecución, la lectura de valores y, desde luego, la información de sus operandos.

Tabla 27. Estructura de una fórmula

CAMPO	TIPO	DESCRIPCIÓN
Número de variable	Entero	Es el número de la variable de esta fórmula
Nivel	Entero	Su valor es 0, después se actualizará.
Factor aplicable	Entero	Es el factor que se aplica a los valores de esta variable.
Subtotal ()	Subtotal	Denota un paréntesis dentro de la fórmula, puede almacenar hasta cuatro operandos en un registro del arreglo, tiene máximo cuatro registros con cuatro operandos posibles en cada uno
Cuántos subtotales	Entero	El número de subtotales que tiene la fórmula
Cuántos sinónimos	Entero	Es el número de sinónimos que tiene la variable de esta fórmula.

El tamaño de esta estructura es relativamente grande (en comparación con las anteriores y más aún por las subestructuras que usa), sin embargo, es necesario que así sea, por la importancia de su contenido y su uso en los procesos de preparación del archivo y en la ejecución. Cada registro de una fórmula contiene un arreglo de otra estructura llamada subtotal.

Tabla 28. Estructura de un subtotal

CAMPO	TIPO	DESCRIPCIÓN
Cuántos operandos tiene	Entero	Es el número de operandos con los que cuenta este subtotal. Mínimo 1, máximo 4
Es distribución	Entero	Es verdadero si este subtotal es una distribución
Operación ()	Entero	Este es un arreglo de máximo 4 valores, en los que cada uno contiene un valor entre 0 y 15 que indica (la clave de) la operación
Operando ()	Operando	Es la unidad básica de la fórmula. En cada subtotal aparece un arreglo de máximo cuatro operandos

Como se explicó en la parte de clientes, un subtotal denota los paréntesis dentro de una fórmula, es por eso que guarda cierto número de operandos dentro de él (máximo 4) y las operaciones que afectan a cada uno de ellos. El arreglo *operación* almacena un valor numérico que indica la operación que debe ser efectuada en el operando del índice correspondiente (ver Tabla 28).

Tabla 29. Operaciones disponibles en la fórmula

NÚMERO	OPERACIÓN	TIPO
0	Ninguna	– –
1	Suma	Aritmética
2	Resta	Aritmética
3	Producto	Aritmética
4	División	Aritmética
5	Raíz cuadrada	Aritmética
6	Cuadrado	Aritmética
7	Mínimo	Comparativa
8	Máximo	Comparativa
11	Media aritmética	Estadística
12	Mediana	Estadística
13	Moda	Estadística
14	Desviación estándar	Estadística
15	Varianza estimada	Estadística

Cuando se elige una operación, en la captura de fórmulas, el programa guarda el número asociado a la misma (ver Tabla 29). Estos números se leen a la hora de la ejecución y se determina la operación a aplicar a cada operando.

El otro arreglo del subtotal es el de la estructura tipo operando, el cual tiene características propias como se muestra en la Tabla 30.

Tabla 30. Estructura de un operando

CAMPO	TIPO	DESCRIPCIÓN
Número de operando	Entero	Es el número de la variable que aparece en este operando, si es sinónimo aquí va el número del sinónimo y si es subtotal, su número
Equivalente de	Entero	Aquí va siempre el número de la variable, este valor nunca cambia.
Usa valores de	Entero	Para el caso de que sea una VMR la que está en el operando, aquí va el número real en FLAG de dicha variable.
Tipo de operando	Byte	VCL, VMR, Sinónimo, Subtotal, Constante, “se usará”
Desfase	Byte	Se refiere al periodo (uno de los 24) con el que se efectúa una operación entre dos variables

No está demás detallar el concepto de *desfase* mostrado en la Tabla 30 ya que tuvo modificaciones importantes respecto a su definición en la versión anterior de FLAG. Bauer (2010) en un inicio implementó esta herramienta para efectuar operaciones desfasadas entre subtotales, sin embargo aquí se modificó para hacer operaciones entre periodos de una variable desfasados con los de otra.

El campo *desfase* toma valores desde -23 hasta 23 porque hay 24 periodos de valores en cada variable. La forma en que se almacena, en la estructura del operando, no es este número directamente, al menos no para los desfases negativos. Esto debido a que el tipo byte no acepta valores negativos, para eso se hace es una transformación de valores: cuando el desfase es negativo, el valor en memoria se convierte en 51 para un -1, en 52 para -2 y así sucesivamente.

En el archivo de fórmulas, tras la generación de las partes 1 y 2, es decir, la estructura *registro base* y el arreglo de estructuras de *fórmulas en orden*, empieza el arreglo de estructuras de tipo *una fórmula;* el tamaño de cada registro es el de la estructura mostrada en la Tabla 27. El acceso a cada parte del archivo se hace idicando el byte preciso desde el que se desea extraer información; del mismo modo se efectúan las operaciones de grabación de datos en el archivo. Esto se hace para ubicar el inicio de las fórmulas, también el de las listas de submodelos y las alarmas, gracias a las variables que fungen como apuntadores en el registro base.

4.1.5. Variables compuestas necesarias para el proceso

Para que el lector entienda la preparación del archivo y con ello los procesos internos que se efectúan, es necesario explicar las variables que se usan para el proceso.

Tabla 31. Variables compuestas usadas en el proceso de preparación del archivo de fórmulas

Variables del Cliente	Variables del Mundo Real	Sinónimos
Tiene fórmula	Núm. variable	Variable
Es de arriba	Original VMR	Tipo
Nivel	Dad	Dad
Pass_nivel	Submodelo	Fórmula to dad
CF_esOper	CF_esOper	Submodelo
Pos_ordenada		Original VMR
Dad		
Fórmula to dad		
Submodelo		

La preparación tiene como fin que el sistema conozca la estructura del modelo, en base a las variables y fórmulas que ha capturado el cliente. Para esto se hace uso de tres variables compuestas, mostradas en la Tabla 31. Las características aquí presentadas se aplican a todas las VCL, VMR y sinónimos generados. De ese modo todo el proceso usa estos arreglos de variables compuestas. Cabe señalar que se almacenan los elementos en los arreglos de acuerdo al número correspondiente (de VCL, de VMR o de sinónimo).

Las características de estas variables compuestas definen su posición en la estructura del modelo. Por eso se explicarán los campos que las componen. Hay campos exclusivos de las variables del cliente, como “tiene fórmula”, “es de arriba”, “pos_ordenada” y “nivel”; éste último es la clave para entender los anteriores; como sólo las variables del cliente pueden ser calculadas (tener fórmula), su nivel puede ser mayor a cero. Cuando esto ocurre, “tiene fórmula” es verdadero y si la variable en cuestión no pertenece a otra fórmula, “es de arriba” también se vuelve verdadero.

Las VMR y los sinónimos siempre son variables de abajo, por esta razón no se indica su nivel porque siempre vale cero, es decir, nunca pueden tener fórmula. El “dad” se refiere a la variable (con fórmula) que usa a la actual como operando. Toda variable cuyo dad vale cero “es de arriba”. El campo CF_esOper (de hecho es “de cuántas fórmulas es operando” pero se abrevió de este modo) se refiere al número de fórmulas en las que aparece dicha variable; este campo, al igual que “pass nivel” son de control, el primero para la generación de sinónimos y el segundo para la detección de bucles (ciclos). “Fórmula to dad” se explicará en forma detallada en el apartado de variables aditivas.

Al inicio de la preparación se leen todas las variables, guardadas en la base de datos, y se mandan a memoria en las variables compuestas anteriores. El índice hace referencia al número de la variable. Esto se decidió así para que el acceso a alguna de ellas sea de forma directa, sin necesidad de hacer una búsqueda. Se generan dos arreglos por cada tipo de variable: dos para VCL y dos para VMR. Esto es necesario puesto que hay dos subtipos de las mismas: variables escalares y aleatorias. El índice de cada uno de los arreglos es el número de la variable módulo 1000,000, puesto que las VCL aleatorias comienzan con 1000,001, las VMR escalares con 2000,001 y las VMR aleatorias con 3000,001.

En el caso de los sinónimos, el índice es el número del sinónimo, puesto que se numeran a medida que se crean. Los datos de variable, tipo y número original (cuando es VMR) hacen referencia a la variable que representa, mientras que los demás son propios del sinónimo. Más adelante se detallará el uso de estos campos conforme se vayan ocupando. Este es un arreglo único, no importa si hace referencia a una VCL, VMR o si es escalar o aleatoria, los datos necesarios ya están contemplados.

La estructura de fórmulas, explicada en la sección anterior, también se usa a lo largo del proceso de preparación. Por eso, las fórmulas se almacenan en memoria al inicio, al igual que las variables, como se explicó arriba. Inicialmente el arreglo contiene las fórmulas tal cual se obtienen del archivo creado por el cliente o como lo almacenó por última vez el proceso de preparación de fórmulas.

4.1.6. Calculo de Niveles

Para entender el cálculo de niveles y submodelos, es necesario poner en contexto el tipo de modelo que es usado por el FLAG, que tiene una estructura tipo árbol, descrita anteriormente precisamente para su uso en este proceso.

Cada nodo que compone el árbol representa a una variable del modelo que ha definido el cliente (VCL o VMR). Las hojas del árbol (nodos que no tienen hijos) en el FLAG reciben el nombre de “Variables de abajo”. Cada rama representa una variable cuyos nodos hijos son los operandos de la fórmula de dicha variable.

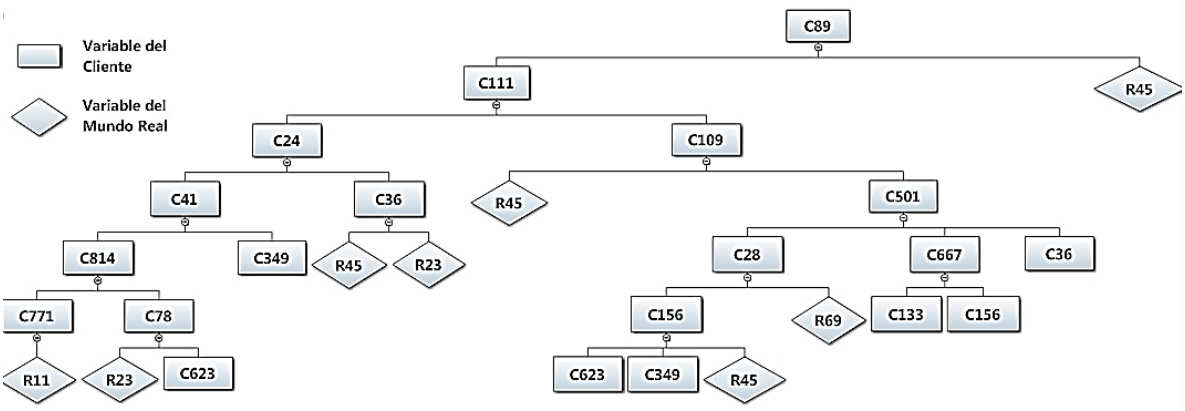


Figura 87. Un ejemplo de modelo para ilustrar el concepto de árbol

En la Figura 87 se muestra el esquema de un modelo: cada nodo representa una variable, hay una variable raíz y sus hijas; algunas de éstas contienen a otras en su fórmula y éstas, a su vez, a otras. El sistema FLAG del cliente ofrece una herramienta con la que se puede diagramar un modelo, y presenta un diagrama semejante al que muestra la Figura 87. Es importante señalar que el diagrama no muestra las operaciones que hay dentro de la fórmula; sólo se indican sus operandos. Para conocer la fórmula a detalle, hay que consultarla como tal en el sistema.

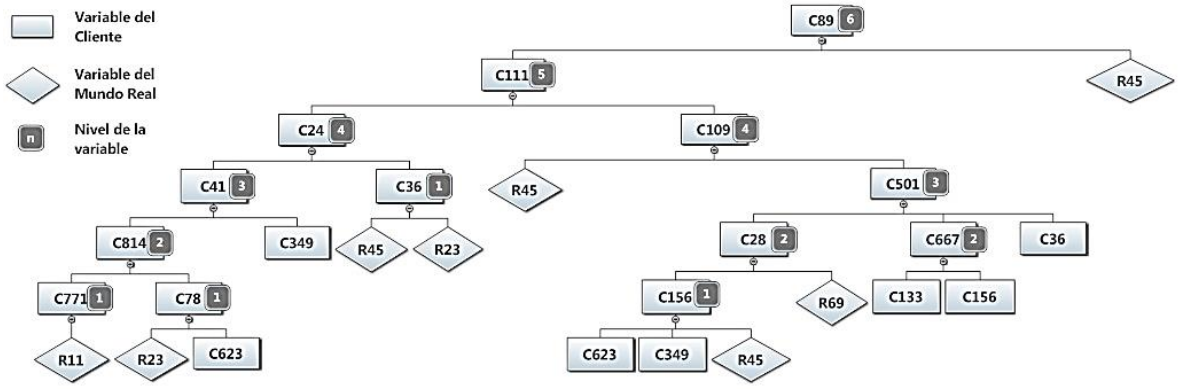


Figura 88. Los niveles de las fórmulas (variables calculadas)

La Figura 88 muestra los niveles de las fórmulas ilustradas en la Figura 87 (es decir, se refiere al mismo modelo). Observe que las variables de abajo tienen nivel 0 y, en este ejemplo, el nivel máximo es 4. A continuación se describe la regla para asignar los niveles, que constituye la definición del nivel: *"El nivel de una variable con fórmula es igual al nivel de su operando de mayor nivel más uno."*

Es decir, nivel (fórmula) = máximo (nivel de sus operandos) + 1

La lógica de la rutina que asigna los niveles a las fórmulas es la siguiente:

1. Inicializa el nivel de todas las fórmulas con un 0 (cero).

Observación*:* Siempre, al capturar (o adquirir) una variable, su nivel por default es cero. En el caso de tratarse de variables calculadas, este valor cambia, por lo mismo se reinicializan con cero al ejecutarse el presente proceso. Para las variables de abajo no hay problema ya que nunca cambia su nivel, que es igual a cero.

2. Recorre todas las fórmulas y para cada una,
    a. Para cada uno de sus operandos, obtiene el nivel del arreglo correspondiente
    b. Determina el nivel máximo de esos operandos
    c. Asigna a la fórmula: el nivel máximo + 1 y actualiza este dato en el uno de los arreglos de VCL (según el tipo de variable, escalar o aleatoria). Observe que de ese modo, cuando esta variable aparezca como operando de otra fórmula, habrá cambiado su nivel.
    d. Si el nivel nuevo de la fórmula no coincide con lo que tenía anteriormente, indica que HUBO CAMBIOS.
3. Si no hubo cambios, termina el proceso.
4. Si hubo cambios, regresa al paso 2.

Circunstancia de excepción:

Si el programa recorre “demasiadas” veces las fórmulas, eso indica que hay un ciclo (*loop*). Para concretar el término demasiadas en el contexto presente, se asigna a una variable “número máximo de recorridos” el valor

Número de fórmulas + 1.

Cuando se han recorrido todas las fórmulas este número de veces, esto indica que se ha entrado en un *loop*. Para evitarlo, se detiene el cálculo de niveles y se procede a informar al cliente.

Un ciclo o *loop* significa bucle. En el modelo se refiere a una *referencia circular* que sucede cuando en una o más variables con fórmula en sus operandos se hace referencia a sí misma, ya sea directa o indirectamente.

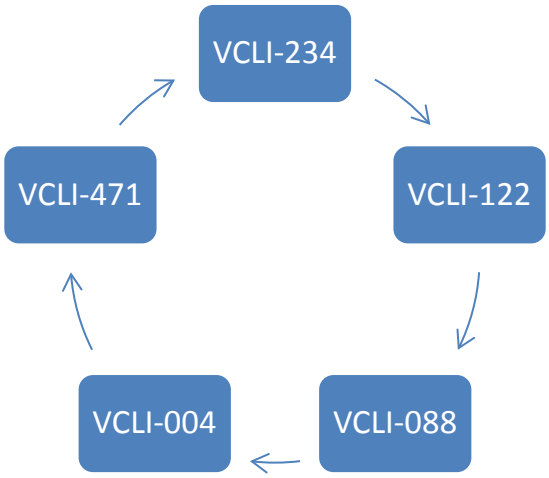


Figura 89. Redundancia cíclica de fórmulas (ejemplo 1)

Las cinco variables de la Figura 89 tienen fórmula, la 234 tiene como operando a la 122, ésta a la 88, ésta a la 4, ésta a la 471 y ésta contiene a la inicial 234. Observe que en este caso no se puede dibujar el modelo, es decir no será un árbol (finito).

Evidentemente, esto surge de una formulación errónea del modelo (o un error en la especificación de la fórmula a la hora de introducirla al sistema).

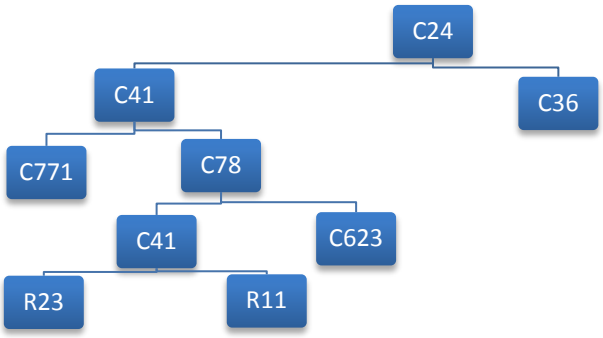


Figura 90. Redundancia cíclica de fórmulas (ejemplo 2)

En la Figura 90 se muestra una parte del modelo (modificado), donde claramente aparece un *loop*. La variable de cliente número 41 aparece como operando de la variable 78, pero también aparece como operando en la fórmula de la variable 24. El proceso de ejecución calcula primero los operandos y después la fórmula que los usa; para este caso no puede calcular la fórmula de la variable número 41 hasta que calcule la de la número 78, pero ésta no será calculada hasta que se calcule la número 41. A este tipo de errores el software Microsoft Excel, por ejemplo, lo llama referencia cíclica, un proceso en el que la variable se refiere a sí misma lo que resulta en un ciclo infinito.

El proceso que asigna nivel a las fórmulas hace uso del arreglo de fórmulas que se generó en memoria y los arreglos de las variables compuestas mencionadas antes. La lógica de asignación de niveles ya fue mencionada antes, sin embargo, la primera vez que recorre las fórmulas se determinan otras cuestiones. La secuencia del proceso es de la forma siguiente:

En la primera “pasada” (lo que en computación se refiere a la lectura de todos los datos de un conjunto en forma secuencial):

a. Se asigna la variable de esta fórmula como dad a cada uno de sus operandos (siempre que no se trate de un subtotal) en su correspondiente elemento de la variable compuesta a la que se refiere.
b. Se especifica, en el operando, que su variable es la original o se trata de un sinónimo. Esto es, cuando ha aparecido más de una vez en el modelo.
c. Se crean los sinónimos. Esto cuando la variable ya existe en otra fórmula (esto se explica con más detalle en la sección de sinónimos).
d. Se determina la aditividad de cada uno de los operandos
e. Se asigna como falso a “es de arriba” a cada operando.

4.1.7. Generación de Sinónimos

El motivo por el que existen los sinónimos en el sistema es que frecuentemente una variable se usa más de una vez en el modelo y sus atributos hacen referencia a la posición en el modelo donde aparece. Por lo tanto una variable “no puede estar” en dos fórmulas diferentes porque, de entrada, el submodelo ya es diferente, sin embargo, esto puede ocurrir si se le crea un sinónimo. Lo cierto es que, en la mayoría de los casos, más de una variable se usará en más de una fórmula y no pueden tener el mismo submodelo. Para resolver esta situación de se concibió el concepto de sinónimo de una variable. La RAE (2014) define el término sinónimo como: dicho de un vocablo o de una expresión que tiene una misma o muy parecida significación que otro. Con lo anterior se puede intuir el uso y las razones de que se incluyeran aquí.

Un sinónimo es muy semejante a una variable, lo único diferente es su posición en el modelo y lo que ello implica. Cuando una variable se usa en más de una fórmula se genera un sinónimo, el número de ellos depende de la cantidad de fórmulas en las que aparezca. En el modelo, los sinónimos son variables adicionales a las que crea al cliente y a las VMR. En la base de datos existe una tabla individual para almacenar los sinónimos; éstos hacen referencia a la variable original pero con un submodelo diferente. Como son sólo referencias, no cuentan con espacio de almacenamiento de valores ya que cuando se utilizan en la

ejecución, basta con saber a cuál variable corresponde (este dato está especificado como campo de cada operando de la fórmula) y de ahí extraer sus valores.

Si la variable aparece en tres fórmulas, tiene 2 sinónimos, si aparece en 6 entonces tiene 5. Cuando se captura una fórmula, el cliente simplemente selecciona las variables que necesita como operando y las agrega. Eventualmente las variables se repetirán. En cada operando siempre aparecen con su número correspondiente, después el programa de *ejecución de niveles*, explicado antes, las registra y determina cuántas y cuáles se convertirán en sinónimos. El cliente nunca sabrá si una variable, dentro de una fórmula, es sinónimo o corresponde a la original.

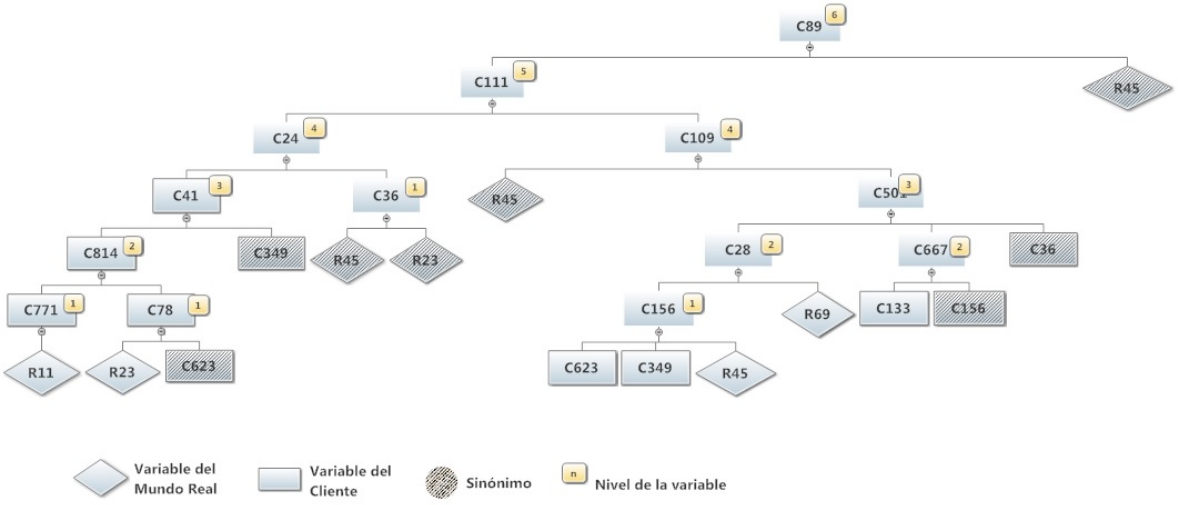


Figura 91. Ejemplo de sinónimos en un modelo

En el modelo de la Figura 91 se pueden ver algunos sinónimos. También se ve que una variable calculada original siempre es la que tiene fórmula y un sinónimo de ésta es la que se usa en otra. Esto solo es visible cuando se dibuja el modelo. Naturalmente, un sinónimo nunca tiene fórmula.

El proceso que genera los sinónimos está incluido en el cálculo de niveles solo que, como se explicó antes, los sinónimos (y otras cosas) se calculan en el primer recorrido o pasada de las fórmulas. El procedimiento que sigue es el siguiente:

- Cada vez que se lee un operando, se registra en cuántas fórmulas aparece la variable del mismo (de hecho, se suma 1 al campo correspondiente);
- Si es la primera vez, el *número de variables en las que aparece* es 1, por lo que en la fórmula de deja la variable original;
- Cuando se procesa otra fórmula en la que también aparezca la variable anterior como operando, el *número de fórmulas en las que aparece* aumenta en una unidad;

- Es a partir de esta circunstancia (el número de fórmulas del cual es operando es mayor que 1) cuando se crean los sinónimos: la fórmula ya no utiliza la variable original sino un sinónimo;
- Al ir detectando variables que ya existen en otra fórmula, se van creando (y numerando) los sinónimos de forma consecutiva sin importar si la variable es de cliente o es una VMR;
- El proceso termina cuando haya recorrido todas las fórmulas del modelo.

Al final se crearán los registros en la tabla de sinónimos en la base de datos, tantos como sinónimos hayan sido creados, con la información mencionada en la Tabla 31. En este punto del proceso, la información que falta en el sinónimo es el submodelo, que más adelante se explica. La demás información ya fue generada en el recorrido de fórmulas, incluyendo la aditividad, concepto que se explica a continuación.

4.1.8. Variables Aditivas

Hay 13 operaciones que involucran a un operando en una fórmula: 5 son estadísticas, 6 aritméticas, dos comparativas y la opción vacía (generalmente se usa al inicio de cada subtotal). Las operaciones que se toman en cuenta para la aditividad son la suma y la resta. Los operandos cuya operación asignada es una de estas dos, son *candidatas* a ser aditivas (pueden ser aditivas, pero no necesariamente lo serán).

En la Tabla 31 se puede ver que los arreglos para VCL y sinónimos tienen un campo “fórmula to dad”, el cual define la aditividad de esa variable o sinónimo, en la fórmula a la que pertenece. Las VMR no tienen con este campo ya que su uso no es necesario por razones que se explicarán en la sección de ejecución.

¿Por qué las variables aditivas?

Este concepto se usa para hacer más eficiente la ejecución. Cuando una fórmula se calcula se deben hacer la operaciones con todos los operandos, esta es la forma común de ejecutarse. Sin embargo, la ejecución de un modelo muchas veces es causada (invocada) por el cambio de valor de una única variable. Si se tiene definida una variable como aditiva dentro de una fórmula, al cambiar sus vaolres se calcula la diferencia del valor nuevo y su valor anterior. Como estos valores intervienen en forma aditiva en la fórmula (del dad) para calcular los valores de ésta basta aplicar dicha diferencia a los valores anteriores. En otras palabras, para calcular los valores de la fórmula del “dad” no hace falta efectuar todas las operaciones (y por lo tanto, no hay que leer los valores de sus operandos). Esto resulta en un ahorro de actividades de lectura de disco, aspecto

importante en especial para las fórmulas que pueden contener hasta trece operandos.

Proceso de asignación de variables aditivas

Una variable aditiva se refiere a las operaciones que afectan al operando que usa dicha variable. Si el operando interviene en una suma o resta y además el operando que le sigue en la fórmula también tiene una de estas dos operaciones, entonces la variable es aditiva. Para el caso en que el operando en cuestión sea final de fórmula o sea el último operando de un subtotal, la variable es aditiva. Cuando un subtotal se usa dentro de otro, la variable es aditiva sólo si el operando, donde aparece el subtotal, también es aditivo. Es importante señalar que si una variable aparece más de una vez en una fórmula, no es aditiva.

La aditividad no está definida para variables aleatorias de las cuales en el operando anterior o siguiente hay también una de éstas. Esto es debido a que las operaciones entre variables aleatorias son muy diferentes a las efectuadas con escalares, pero también porque los periodos de las aleatorias tienen cuatro valores probables y la suma y resta no procede como en una variable escalar y mucho menos con otra aleatoria. Sin embargo, cuando ambos operandos (anterior y siguiente) son escalares, la variable aleatoria se puede convertir en aditiva, claro, siguiendo el criterio explicado arriba.

La aditividad de un operando se indica usando un campo tipo número byte. SE se trata de un sumando, el valor del campo es 1; si es sustraendo, vale 2, y si no es aditivo, el campo vale 0. Este dato se almacena en memoria, en los arreglos de variables compuestas. Ahí se queda hasta que se usa después por el programa que genera las listas de fórmulas. Después se guarda en el archivo y más adelante esta información la usa la ejecución de submodelos (como se verá, cuando se ejecuta un modelo completo no se hace uso de esta característica).

4.1.9. Generación de Submodelos

Este es otro atributo que tienen todas las variables y depende de su posición en el modelo; es decir, cada variable está en un submodelo.

En esta parte se puede entender mejor el uso de los sinónimos, es decir, una variable que se usa en dos fórmulas, en la primera aparece la original y en la otra su sinónimo pero con un submodelo diferente. Tras la ejecución de *niveles y submodelos* a todas las variables, sin excepción, se les asigna un submodelo que además es diferente entre cada una de ellas.

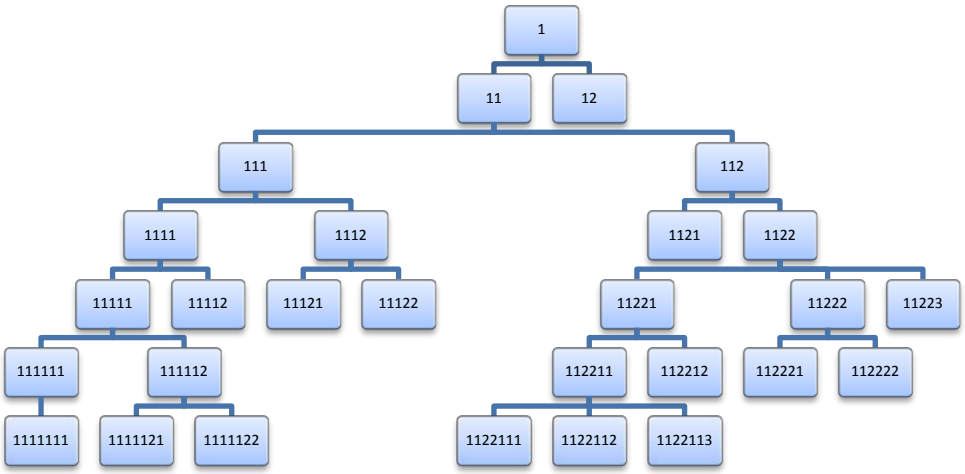


Figura 92. Submodelos de las variables de un modelo

Los submodelos se usan principalmente para dibujar el modelo del cliente. En la Figura 92 se puede ver el ejemplo de un modelo como estructura de árbol, desde el nodo raíz hasta las hojas inferiores. A cada hoja le corresponde una variable (o sinónimo). Cuando los submodelos ya están generados, el modulo que dibuja el modelo hace la representación del mismo en un objeto llamado *treeView* donde se pueden ver, ya sea, los números de variable, los submodelos o los nombres de cada una, dependiendo lo que se quiera mostrar. Como se ha comentado, también hay un programa de FLAG que “dibuja” el modelo y lo muestra en el monitor. El cliente puede invocar la descripción de la variable correspondiente con un click de su ratón. Cuando se trata de un sinónimo, se le informa cuál variable representa.

Proceso que genera los submodelos

Para iniciar el proceso que genera los submodelos primero se deben asignar los niveles, crear sinónimos y definir variables aditivas, según sea el caso.

- Este proceso inicia recorriendo las fórmulas que son de arriba, es decir, las que tienen como dad 0.
- De cada una de ellas se desprende una estructura de árbol como la de la Figura 87 con variables y otras fórmulas.
- A cada variable de arriba se le da un número consecutivo de submodelo, iniciando desde el 1 y máximo 9.

Observación: si por alguna razón el cliente capturó mal el modelo y resultan más de 9 variables de arriba, los submodelos no se generarán para las variables después de la número 9. En muchos modelos el cliente tendrá una única variable de arriba, la más importante que él necesita conocer. Puede haber otras variables *importantes* pero no necesariamente son de arriba. No se

descarta que se lleguen a usar los nueve submodelos pero el cliente debe estar consciente de lo que está manejando.

- Una función recursiva recorre todos los operandos de cada fórmula. A cada variable del operando le asigna un nuevo submodelo con números consecutivos agregados al final del anterior submodelo.
- Hace lo mismo con los operandos cuyas variables también tienen fórmula.

Esto se entiende mejor analizando la Figura 92. El submodelo indica el camino que se debe seguir para llegar hasta la variable de arriba que afecta una variable. Esto quiere decir que entre más *largo* (número de caracteres) sea el submodelo de una variable, mayor número de fórmulas son afectadas por el cambio de ésta.

4.1.10. Almacenamiento de Fórmulas

Los procesos anteriores son necesarios para que los siguientes sean posibles, ya que con éstos se genera la información de las variables compuestas y también se actualizan los campos *tipo de operando* y el *dad* de las fórmulas. Pero antes de proseguir con la creación de listas de submodelos (fórmulas a ejecutar) y la incorporación de los criterios de alarma, es necesario actualizar la información que hay en el archivo plano de fórmulas, puesto que estos segmentos preceden a los submodelos en el archivo de fórmulas preparadas. La estructura general de la información que se guarda en el archivo de fórmulas, está representada en la Tabla 32.

En el registro base solo cambia el dato de “dónde empiezan los submodelos” ya que el número de fórmulas no cambia y los demás campos se modifican con los procesos que siguen. La lista ordenada de fórmulas sí cambia y hay que generarla nuevamente en base a los niveles calculados. Las fórmulas se graban una tras otra en el orden que indica la lista ordenada.

Tabla 32. Esquema de la composición del archivo plano de fórmulas.

1	2			3			4			5
REGISTRO BASE	**VARIABLES ORDENADAS**			**FÓRMULAS**			**SUBMODELOS**			**ALARMAS**
	VAR1	VAR2	VAR3	FÓRM1	FÓRM1	FÓRM1	INDEX	LIST1	LIST2	CRITERIOS

Proceso que guarda fórmulas antes de la Ejecución de Niveles y Submodelos

El archivo se crea cuando se inicia la captura de la primera fórmula: se crea en memoria la estructura *registro base* con datos iniciales, la mayoría en cero y las fechas con el dato de la fecha del día. La lista de variables ordenadas se inicia con tamaño 100 (si hace falta, se redimensiona este arreglo cuando aparezcan

fórmulas adicionales a la dimensión anterior). El número de elementos que se usan de la lista inicia con 1, es decir, el de la fórmula que se está procesando en este momento. Este número irá aumentando a medida que se lean las fórmulas subsecuentes.

Cuando ya no hay fórmulas (se leyó la última) se redimensiona la lista al número correcto y se actualiza el valor, en el registro base, de la cantidad de fórmulas existentes. Las fórmulas se guardan después de estas dos estructuras, una tras otra conforme se van leyendo. El *registro relativo* (ver Tabla 26) de la estructura de *variables ordenadas* es el consecutivo donde se va guardando la fórmula.

Proceso que guarda fórmulas después de la Ejecución de Niveles y Submodelos

Las rutinas que se encargan de completar el proceso son las siguientes:

- *Arma arreglos por nivel*. Se recorre el arreglo de variables con fórmula, se crean arreglos temporales para guardar las que pertenecen a cada nivel y la cantidad que hay en cada uno.
- *Arma lista a grabar*. Se recorren los arreglos de la rutina anterior, las variables se guardan en *variables en orden* (que está en memoria) desde el arreglo de variables con nivel 1 hasta el máximo.
- *Guarda archivo nuevo*. Se recorre el arreglo *variables en orden* ya actualizado y se guardan las fórmulas en el archivo en el mismo orden.
- *Actualiza la base de datos*. La información que se ha obtenido en todo este proceso se guarda en la base de datos. En las tablas de variables de cliente, variables del mundo real que usa el cliente y sinónimos se guarda la información de niveles, submodelos, *dad y fórmula to dad* que se encuentra en las variable compuestas en memoria. De este modo las variables están actualizadas para consulta del cliente.

Observación. Esta técnica para armar la lista de fórmulas en orden fue resultado de una decisión de programación. Se podía haber usado otro algoritmo: agregar cada una de las fórmulas a la lista pero en la posición indicada por su nivel (en orden ascendente del mismo).

Al terminar de guardar las fórmulas se actualiza un valor del *registro base* llamado *dónde empieza lo de submodelos*, este valor es el número de byte donde comienza la información de la lista submodelos que es igual al tamaño actual del archivo más un byte.

4.1.11. Listas de Submodelos (partes 4 y 5 del archivo)

La creación de listas de submodelos o listas de fórmulas a ejecutar es una de las partes más importantes del servicio FLAG. Éstas se concibieron con el fin de agilizar la lectura de fórmulas a la hora de la ejecución. Teniendo en cuenta que el servicio consta de mantener informado al cliente que hace uso de variables del mundo real proporcionadas por FLAG y hacerlo de forma correcta y en tiempo real, es imperativo que las ejecuciones sean lo más eficientes posibles. Esto se logra con las listas de fórmulas a ejecutar para cada VMR, es decir, las que son afectadas por esta VMR cuando ha cambiado. Aparte de reducir la cantidad de fórmulas que se ejecutan, la aditividad aporta más eficiencia al cálculo de cada fórmula, como se verá más adelante.

Tabla 33. Lista de fórmulas por submodelo

Campo	Tipo	Descripción
Número de la variable	Entero	VMR de este submodelo
Número de la variable (para el cliente)	Entero	Número con el que el cliente conoce e esta VMR
Cantidad de fórmulas en el submodelo	Entero	
Posición de la lista	Entero	Apuntador a la ubicación en el arreglo de listas de fórmulas

Tabla 34. Estructura índice de submodelos

Campo	Tipo	Descripción
Número de variable	Entero	Una variable con fórmula, obviamente
Posición de su fórmula	Entero	Apunta a la ubicación de esta fórmula en la parte 3 del archivo
Hija	Entero	Variable que invocó la ejecución de esta fórmula (si dicha variable es un operando aditivo)
Nivel	Entero	Nivel de la fórmula

La parte 4 es un arreglo tipo índice, semejante al de la lista de variables ordenadas en la parte 2, pero ahora con la estructura de la Tabla 34. La parte 5 del archivo es un arreglo de arreglos de la estructura de la Tabla 33, donde cada sub arreglo se refiere a una lista de fórmulas.

La definición de un submodelo es la posición en la que, una variable o sinónimo, se encuentra en el modelo. Por ejemplo, una variable con submodelo 2251 está en la fórmula de la variable cuyo submodelo es 225 y ésta forma parte de la de submodelo 22. Este es el camino que se sigue para generar la lista de variables con fórmula que son afectadas por el cambio de una VMR.

Pero ocurre que la VMR con submodelo 2251, por ejemplo, puede tener varios sinónimos con otros submodelos diferentes al de la variable. Para cada uno de ellos se efectúa el proceso anterior buscando, en orden ascendente, las variables

del modelo que son afectadas por el cambio. Pero el proceso no acaba ahí porque las variables con fórmula que son afectadas también pueden tener sinónimos y, por consecuencia, otro submodelo y otras variables que también son afectadas. La rutina encargada de este proceso lo hace de forma recursiva, primero las variables afectadas y después lo mismo para cada sinónimo de cada variable afectada, iniciando con la VMR en cuestión.

Cuando se leen las listas como parte de un proceso de ejecución del submodelo, las variables ya están ordenadas por nivel, de modo solamente debe ubicar la lista (por su apuntador en el arreglo de submodelos) y ejecutar cada fórmula en el orden especificado.

El proceso comienza leyendo las VMR que hay en el modelo, eligiendo únicamente las que están en por lo menos una fórmula como operando. Las VMR con *dad* igual a cero se descartan (en general no habrá tales, pero un cliente puede decidir incluir una tal variable simplemente para conocer su valor, sin que éste afecte otras variables del modelo).

Esta información ya está en el arreglo de estructuras de variables que se usó en los procesos anteriores, es decir, la información de los *dads*, niveles y posición de las fórmulas ya está disponible y puede ser usada en este proceso. La cantidad de fórmulas y la posición se inicializan en cero.

Tabla 35. Estructura “dads de los sinónimos”

Campo	Tipo	Descripción
Número de variable	Entero	Variable con fórmula a la que hace referencia
Nivel	Entero	Nivel de la fórmula
Formula to dad	Byte	Aditividad del sinónimo
Posición de la fórmula	Entero	Posición en la que fue guardada esta fórmula

La estructura “dads de sus sinónimos” de la Tabla 35, se maneja como un arreglo temporal, donde se irán almacenando las variables en cuyas fórmulas aparecen los sinónimos de cada variable afectada por el cambio de la VMR.

El procedimiento que genera estas listas de fórmulas es el siguiente:

1. Se lee la variable compuesta de las VMR, que está en memoria, para determinar una por una las VMR para las que se debe construir una lista. Aquí se llena la lista índice de variables, que es la parte 4 del archivo.

2. Para cada VMR se buscan sus sinónimos en el modelo mediante una rutina específica para ello. En el arreglo “dads de los sinónimos”, a excepción de la posición 0, se guardan las variables que contienen en sus fórmulas a estos

sinónimos. En la primera se guarda el dad de la variable original con sus correspondientes datos: nivel, aditividad y posición de la fórmula.

3. Para cada variable del arreglo de sinónimos se hace lo siguiente:
   a. Se agrega la variable con fórmula de la posición 0 a la lista de fórmulas de la actual VMR (del tipo de la Tabla 33).
   b. Se determina el valor del campo “Hija”. Esto se determina dependiendo si la actual VMR es o no aditiva en la fórmula. Si aparece sumando se guarda la variable hija con signo positivo; si está restando a la fórmula, se guarda con signo negativo; y si no es aditivo tendrá valor 0. En este primer paso la hija siempre es una VMR.
   c. Se procesan todos los dads afectados por la VMR, es decir, el de la original y los de sus sinónimos.
   d. A continuación se ejecuta la rutina recursiva. Ésta recibe como parámetro a la variable que debe ser procesada (un dad de los anteriores mencionados). Se buscan sus sinónimos con la misma rutina de antes y se guardan en otro arreglo de tipo “dads de los sinónimos”. Igual que antes, en la posición 0 se almacena la variable de la fórmula que contiene a la original. El proceso siguiente es el mismo para cada uno de estos nuevos dads:
      i. Se agrega la variable con fórmula de la posición 0 a la lista de fórmulas de la actual VMR (del tipo de la Tabla 33).
      ii. Se determina el campo “Hija”. Este parámetro es la variable que entró en la rutina recursiva como “dad que se procesa”.
      iii. Se determina su aditividad, dependiendo del valor guardado en “fórmula to dad” en su respectiva variable compuesta. De la misma manera que antes, se agrega un signo “+” o un “-” al valor de la variable hija.
      iv. Si la variable ya existe en la lista de fórmulas a ejecutar, no se agrega a la lista y la aditividad tampoco tiene efecto en la ejecución de esa variable (si tenía una indicación de aditividad, es decir, hija no es cero, se quita esta indicación). Si no existe en el arreglo, se la agrega pero en el orden indicado por el nivel, y se determina la hija que lo invocó, si aplica. En este paso la variable hija es siempre una del cliente.
      v. Se procesan los dads de los sinónimos que se obtuvieron de la variable que llegó como hija en la rutina recursiva. La rutina que procesa a cada uno de éstos es la que se está usando actualmente, esta es la parte en que se llama a sí misma. Se manda como valor del parámetro cada uno de los dads de los sinónimos y se vuelve a hacer el proceso desde el paso C.

Un programa recursivo, si no se tiene cuidado, puede convertirse en un proceso infinito. Para este caso la sentencia que define el término del proceso es cuando, en la rutina recursiva, se recibe como hija el valor 0. Esto quiere decir, hasta que una variable procesada tenga como *dad* el valor de 0, por ser variable de arriba. Sin embargo, aún están los dads de sus sinónimos que continuarían la ejecución. El proceso completo termina hasta que la última variable, la de arriba, ya no tiene sinónimos.

De esta manera se han creado listas de fórmulas, una para cada VMR del modelo del cliente. Lo siguiente es almacenar la información obtenida al archivo de fórmulas, primero la lista índice de submodelos y después, cada una de las listas.

4.1.12. Otra explicación de la construcción de las listas de submodelos

Se presenta para conveniencia del lector otro modo de explicar el proceso que crea las listas de fórmulas de un submodelo, basada de un ejemplo.

Se elaboró el modelo que ilustra la Figura 93, preparado con el único fin de ilustrar el proceso de creación de un submodelo, de modo que no es necesaria la interpretación de las diversas variables como parte de una situación real. Un signo "+" o "-" en los conectores de las variables con sus "dads" indica que esa variable interviene en forma aditiva en la fórmula de dicho padre. Se describe la creación del submodelo de la VMR **R1** que se puede resumir del siguiente modo: a) se ubican los sinónimos de R1; b) se construye un arreglo inicial con el dad de R1 y de sus sinónimos, es decir, C12 (el dad de R1) y C16 (el de su único sinónimo); c) cada uno de estos dads se agrega a la lista de fórmulas y, a su vez, se invoca una subrutina que arma el arreglo de los dads de los sinónimos de la variable que llega como parámetro. Este último proceso es recursivo: para cada uno de los dads de dicho arreglo, se hace lo mismo que en el original.

En la Figura 94, que describe el proceso de armado del submodelo, se usó la siguiente notación:

Arr_dads: los "dads" de los sinónimos de la variable. Para cada fórmula de la lista se incluye el número de variable, el nivel de la fórmula y la hija que invoca dicha fórmula.

➔ L: agregar esta fórmula a la lista de fórmulas del submodelo

SRR(Ci, H) invoca la subrutina recursiva con los argumentos: Ci, la variable; H, el número de la variable que invocó la ejecución de Ci; si H no es operando aditivo, se pone 0 (cero); si es sumando, se usa el número de la variable, si es restando (sustraendo) se cambia el signo del número de la variable.

En el diagrama se muestra la lista de fórmulas conforme se agregaban al submodelo, esto es, en el orden en el que se generaban. También se agregó una lista final de fórmulas, la cual debe tener las siguientes características:

- Sólo se especifica la hija si ésta interviene en modo aditivo en la fórmula que la invoca (de lo contrario, se indica hija = 0). Como ya se explicó, las VMR no se consideran aditivas;
- Las fórmulas deben estar en orden ascendente de nivel;
- Si una fórmula está repetida, se elimina la repetida y se indica “hija = 0”. Se podría invocar dos veces el cálculo si ambas tuvieran hijas aditivas, pero se decidió no contemplar esta situación (ver C18 en el diagrama de la Figura 93).

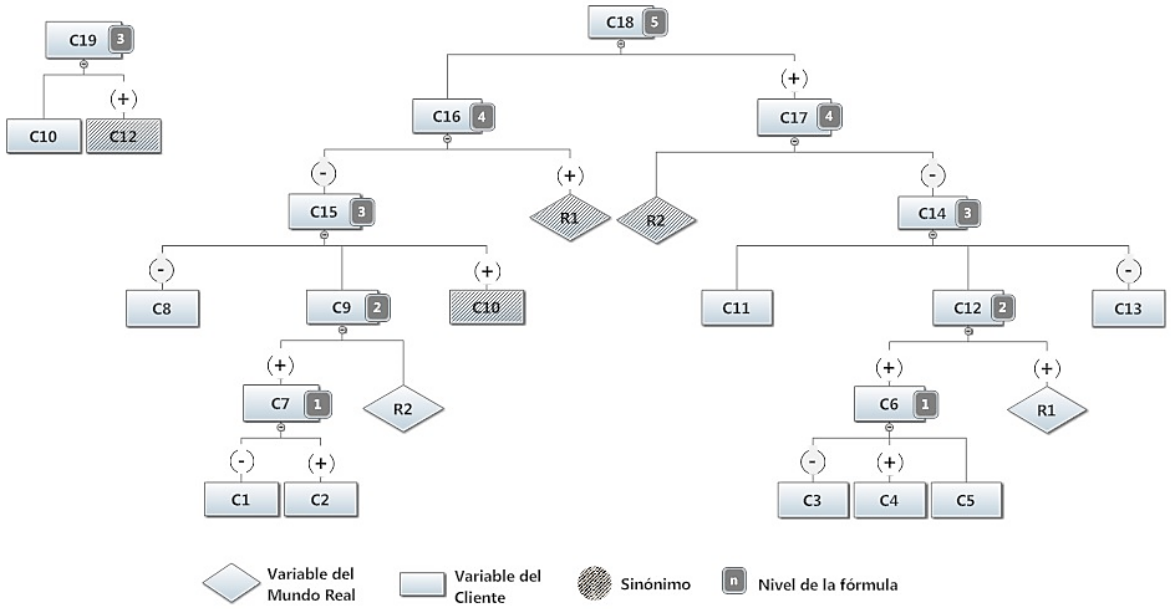


Figura 93. Modelo utilizado para ilustrar el algoritmo que prepara un submodelo

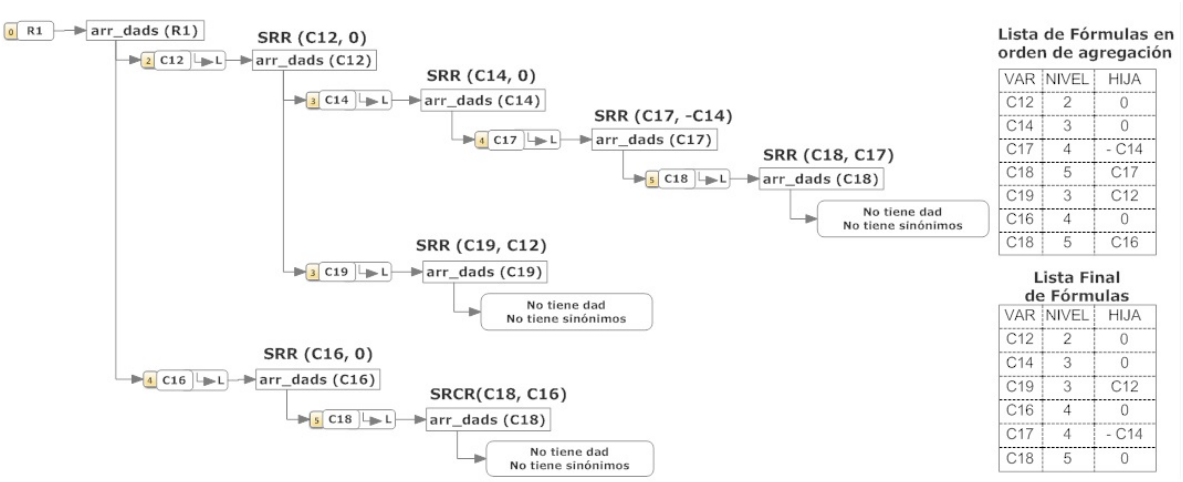


Figura 94. El algoritmo que construye la lista de fórmulas del submodelo de la VMR R1 y la lista fórmulas que produce.

4.1.13. Incorporación de Criterios de Alarma (partes 6 y 7 del archivo)

Cuando el cliente asigna los criterios de alarma, se guardan en tablas de su base de datos. Para su uso durante la ejecución del modelo (el motivo por el cual los define) se tomó la decisión de almacenarlos en el mismo archivo de fórmulas. La ejecución obligatoriamente debe leer la fórmulas del modelo, donde sea que éstas se encuentren; el hecho de estar en un archivo plano ya es una ventaja porque es más rápido que otras opciones y se accede a ellas una sola vez en el proceso.

Los criterios que están en la base tienen un formato específico con el que se guardan. Cuando se exporta esta información al archivo se hace una transformación para adaptarla a estructuras que hacen más eficiente la lectura. Si el cliente decide no crear alarmas en su modelo, la parte del archivo que corresponde a éstas no se genera.

En general, el proceso consta de lo siguiente: leer la base de datos, guardar la información en memoria, transformarla (pasarla a otras estructuras) y, finalmente, guardarla en el archivo plano.

La parte número 6 del archivo almacena información que se usa en la generación del aviso para este cliente, una vez ejecutadas las alarmas.

Tabla 36. Información general de alarmas

Campo	Tipo	Descripción
E – mail	Texto	Información personal del cliente
Teléfono	Texto	
Nombre	Texto	
Variables con alarma	Entero	Cantidad de variables con alarma en este modelo
Puntos de cada rango (4)	Entero	Arreglo de 4 elementos que determinan el tipo de aviso que se generará

Esta parte no corresponde a un arreglo, sino que la estructura siempre es de tamaño fijo. Simplemente se lee la información de la base de datos y se deposita en una estructura del tipo mostrada en la Tabla 36.

Cada variable con sus criterios de alarma se almacena en la parte 7 del archivo a manera de arreglo de la estructura mostrada en la Tabla 38.

Tabla 37. Criterios por periodo

Campo	Tipo	Descripción
Intervalos	Byte	0: no tiene criterios; 1: sólo usa el intervalo crítico; 2: usa crítico y emergencia
Límite inferior (2)	Entero	Valor mínimo de cambio en la variable que se toma como significativo
Límite superior (2)	Entero	Valor máximo de cambio en la variable que se toma como significativo
Puntos (2)	Byte	Cuando el cambio en la variable es significativo, el puntaje aquí almacenado se acumula en otra variable

Los límites, así como el puntaje, son arreglos de dos elementos: el primero corresponde al rango crítico y el segundo al de emergencia.

Tabla 38. Variable con alarma

Campo	Tipo	Descripción
Número	Entero	Variable a la cual corresponde esta alarma
Nombre	Texto	Nombre de la variable
Shortie	Texto	Nombre corto de la variable
Tipo Límites	Byte	Formato de comparación de límites: absoluto, amplitud o porcentaje
Tipo periodo	Byte	Define el tipo de selección de periodos con alarma
Periodo cero	Texto	Variable de control para lectura de periodos en la ejecución
Parámetros por periodo (24)	Criterios por periodo	Arreglo de 24 elementos donde cada uno almacena la información mostrada en la Tabla 37

La información de los tipos de límite y periodo se explicaron en el capítulo de clientes mientras que el periodo cero y la estructura de criterios por periodo se detallarán más adelante en esta sección.

En la Tabla 38 se muestra la información que debe tener cada variable con alarma para servir a las rutinas del programa de ejecución. Por ejemplo, se debe saber si la forma de comparar es un porcentaje o número absoluto; mientras que el tipo de periodo dice cuántos de estos tienen definida una alarma; el periodo cero muestra cuáles son dichos periodos y, por último, está la subestructura de los parámetros o criterios que hay en cada uno de los periodos. Procede el mismo comentario aplicable a otros archivos del FLAG: se diseñaron contemplando su eficiencia de uso por los programas que necesitan la información, lo que resultó en que son un tanto *complicados;* sin embargo la comprensión del sistema FLAG no exige conocer los detalles de estos archivos.

La tabla de la base que se lee primero es la de “CLIENTE”, que contiene los datos de su e-mail, número telefónico de contacto y valor de puntajes para evaluar el envío de avisos. También está la cantidad de variables con alarma junto con el tipo de acumulación de puntos: este dato se usa en la ejecución.

Hecho lo anterior, se prosigue a leer la tabla *alarmasVariables* de la base, donde están los datos principales de cada variable que indicó como crítica (es decir, sobre la cual formuló un criterio de alarma). Se recorre cada registro y se guardan en la estructura *variables con alarma*, en memoria, los campos nombre, número de la variable, descripción, tipo de periodo y tipo de límites. Después, en la misma rutina se invoca otra que leerá la tabla *alarmasVariables_periodos* para obtener los límites y puntajes y almacenarlos en la variable *parámetros críticos alarma* de la estructura antes mencionada.

La tabla *alarmasVariables* contiene las variables que cuentan con alarma, un registro por cada una, sin repetirse ya que el campo variable funciona como clave

principal. En cambio, en la tabla *alarmasVariables_periodos* las variables sí pueden repetirse porque una variable puede tener hasta 24 periodos con un criterio diferente. La clave principal está formada por los campos: variable y periodo inicial.

El tipo de periodo guardado en la tabla *alarmasVariables* define la cantidad de registros en la tabla para esa variable. Sólo se almacena más de un registro cuando es de tipo 4 o tipo 7, es decir, una lista de periodos.

La estructura que guarda la información de cada variable con alarma contiene un arreglo de estructuras de tamaño 24, y tiene por nombre *parámetros_crit_alarma*, donde quedará la información de cada uno de los criterios de cada periodo. Aunque se mencionó que en sólo dos tipos de periodo con alarma se tiene más de un registro almacenado en la tabla, en memoria no será así ya que se utilizarán los 24 de ser necesario. A continuación se resume la forma en que se almacena la información de los periodos en cada uno de los tipos que hay:

1. ***Uno solo***. En la base hay un solo registro que corresponde al periodo que fue indicado. En memoria se guarda también uno solo en la posición del arreglo *parámetros críticos alarma* que le corresponde al número de periodo que está definido por el campo “periodo inicial” de la tabla. Las demás posiciones permanecen vacías.
2. ***Todos los periodos***. Cuando se define una alarma para todos los periodos también hay un solo registro, esto porque el criterio es el mismo para todos. En memoria se guarda el criterio en cada una de las 24 posiciones del arreglo.
3. ***Rango***. El rango consta de un criterio único, almacenado en un solo registro, que se aplica a un determinado número de periodos definido por los campos “periodo inicial” y “periodo final”. En memoria el criterio se guarda solamente en los periodos comprendidos en el rango pero ahora en su correspondiente posición del arreglo.
4. ***Lista***. En este caso sí hay varios registros con diferente criterio. En cada registro el periodo inicial es igual al final, así que se lee cada uno y se manda a memoria en su respectiva posición en el arreglo. Los periodos que no se utilizan quedan vacíos.
5. ***Actual***. Esta alarma está definida sobre un solo periodo, el actual; el criterio se guarda en un solo registro. Ya que el periodo indicado es variable, el criterio se guarda siempre en la posición 0 del arreglo.
6. ***Rango (referente al actual).*** En este caso también es un solo criterio para el rango definido; el detalle es saber cuáles serán los periodos afectados si el periodo actual es variable. Lo que se hizo primero es encontrar la cantidad de periodos que será afectado y guardar el criterio

en ese mismo número de posiciones. Por ejemplo, si el rango es de 6 periodos, se guarda el criterio desde la posición 0 hasta la 5. La cantidad de periodos se obtiene restando del periodo final el inicial. También aquí se hace uso del campo *periodo_cero* de la variable con alarma, donde se guarda el periodo inicial del rango en formato string. Esto es para que la ejecución sepa a qué periodos debe aplicarle el criterio.

7. ***Lista (referente al actual).*** Este tipo de alarma para periodos es el más complejo de todos porque se definen criterios diferentes (uno en cada registro) y los periodos son variables respecto al actual, es decir, no siempre se aplicarán a los mismos, como ocurre en el caso anterior. Se determina primero la cantidad de periodos que serán afectados por el criterio, esto lo da el número de registros en la tabla *alarmaVariable_periodo*. Igual que antes se guardan los criterios a partir de la posición 0 del arreglo. En este caso no es suficiente guardar un solo dato en la variable *periodo_cero* ya que la lista no es necesariamente consecutiva, por eso se toman los valores “periodo inicial”, se convierten a tipo *string* y se guardan en forma consecutiva en *periodo_cero*, separados por una diagonal uno de otro. Así entonces la ejecución sabrá cuáles son los periodos afectados.

Cuando ya se tiene la información de los criterios en memoria se procede a insertarlos en el archivo plano. En la posición *dónde empieza lo de alarmas* se guarda la estructura de *información general de alarmas* y después el arreglo de estructuras de tipo *variables con alarma* que, a su vez, contiene otro arreglo de tipo *parámetros críticos alarma* donde están los criterios de cada periodo.

4.2. Variables del Mundo Real y su Modelo

Las variables que el servicio FLAG pone a disposición de todos los clientes son las del mundo real (VMR). Éstas, como ocurre con las de los clientes, tienen dos clasificaciones: por tipo de valores que manejan y por la forma en que los obtienen. Los tipos de valores son dos: escalares y aleatorios. Por la forma en que se obtienen están las variables de abajo y las calculadas.

El sistema genera un modelo semejante al que se genera con los clientes, las diferencias son que aquí se usan solo VMR y la obtención de los valores de las VMR de abajo es por medio de agentes de información. La estructura del modelo también es de tipo árbol, por las fórmulas que se generarán eventualmente. Sin embargo, el modelo, en teoría, no será tan complicado como puede llegar a ocurrir en uno de algún cliente. Las fórmulas calcularán valores para otras VMR que no dependen - directamente - de agentes y se pondrán a disposición de los clientes. En especial, se podrán generar variables aleatorias como, por ejemplo, de datos

climáticos (precipitación, evaporación, temperaturas), tendencias en producción de hortalizas, comportamiento de la Bolsa Mexicana de Valores, así como otros índices financieros, etc.

4.2.1. Abc de Variables del Mundo Real

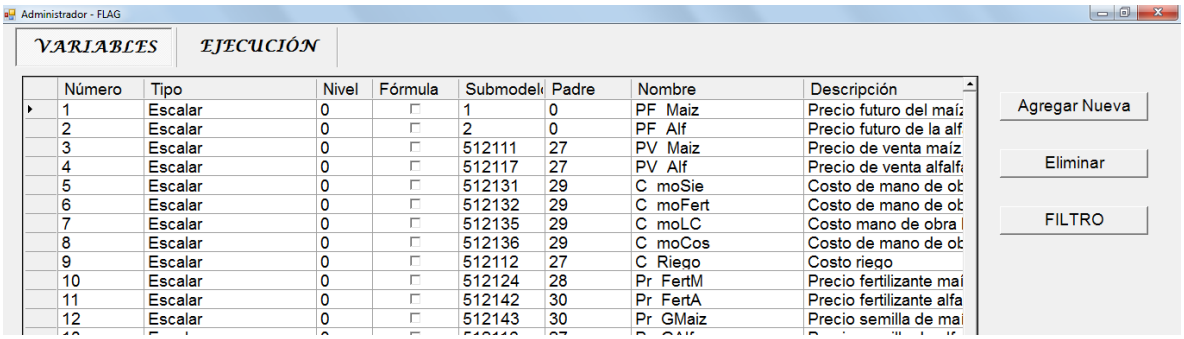


Figura 95. Interfaz de variables del mundo real

El sistema FLAG de variables del mundo real consta de dos grandes componentes: sus variables y el módulo de ejecución. El modelo de las VMR se genera cuando se introducen las VMR a las que se pueden (optativamente) agregar fórmulas. La descripción de este componente del sistema inicia detallando el módulo de variables, la forma en que se realizan las altas, bajas y cambios en ellas. Más adelante se explicará lo de valores y fórmulas y, en la sección de agentes, la forma de asignarlos a VMR que no tienen fórmula, esto para actualizar sus valores.

Primero se debe indicar el tipo de variable que se quiere dar de alta, las opciones son las mostradas en la Figura 96. Seleccionado el tipo de variable a agregar, se “acepta” en la forma y la nueva variable creada aparece en la lista con el número consecutivo al mayor existente de tipo que se creó, en este caso una escalar.

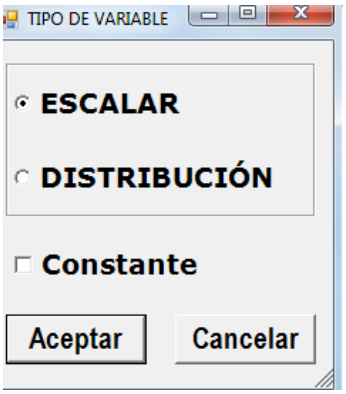
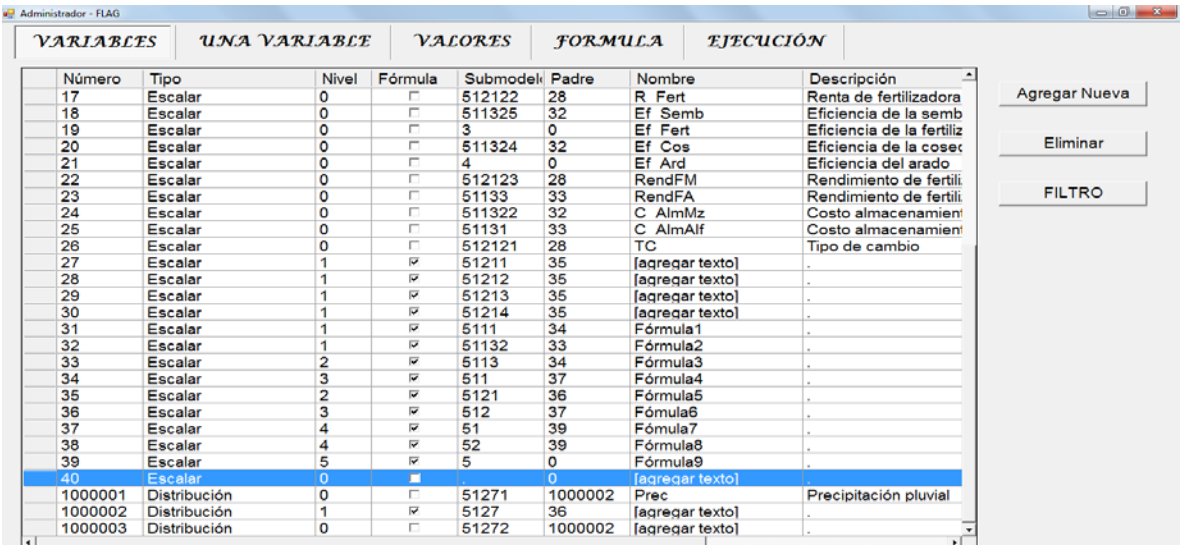


Figura 96. Tipo de variable a agregar

La interfaz inicial de variables se muestra en la Figura 95, donde se aprecian los atributos de cada una, entre los que están el tipo, descripción, shortie o nombre abreviado, unidades, entre otros; también los atributos concernientes al modelo como el nivel, padre (dad), submodelo, fórmula. Para dar de alta una nueva variable se presiona el botón “Agregar nueva”.



Número	Tipo	Nivel	Fórmula	Submodel	Padre	Nombre	Descripción
17	Escalar	0	☐	512122	28	R_Fert	Renta de fertilizadora
18	Escalar	0	☐	511325	32	Ef_Semb	Eficiencia de la semb
19	Escalar	0	☐	3	0	Ef_Fert	Eficiencia de la fertiliz
20	Escalar	0	☐	511324	32	Ef_Cos	Eficiencia de la cosec
21	Escalar	0	☐	4	0	Ef_Ard	Eficiencia del arado
22	Escalar	0	☐	512123	28	RendFM	Rendimiento de fertili
23	Escalar	0	☐	51133	33	RendFA	Rendimiento de fertili
24	Escalar	0	☐	511322	32	C_AlmMz	Costo almacenamient
25	Escalar	0	☐	51131	33	C_AlmAlf	Costo almacenamient
26	Escalar	0	☐	512121	28	TC	Tipo de cambio
27	Escalar	1	☑	51211	35	[agregar texto]	.
28	Escalar	1	☑	51212	35	[agregar texto]	.
29	Escalar	1	☑	51213	35	[agregar texto]	.
30	Escalar	1	☑	51214	35	[agregar texto]	.
31	Escalar	1	☑	5111	34	Fórmula1	.
32	Escalar	1	☑	51132	33	Fórmula2	.
33	Escalar	2	☑	5113	34	Fórmula3	.
34	Escalar	3	☑	511	37	Fórmula4	.
35	Escalar	2	☑	5121	36	Fórmula5	.
36	Escalar	3	☑	512	37	Fórmula6	.
37	Escalar	4	☑	51	39	Fórmula7	.
38	Escalar	4	☑	52	39	Fórmula8	.
39	Escalar	5	☑	5	0	Fórmula9	.
40	Escalar	0	☐	.	0	[agregar texto]	.
1000001	Distribución	0	☐	51271	1000002	Prec	Precipitación pluvial
1000002	Distribución	1	☑	5127	36	[agregar texto]	.
1000003	Distribución	0	☐	51272	1000002	[agregar texto]	.

Figura 97. Nueva variable agregada

En la Figura 97 se puede ver que la VMR creada es la número 40; tras seleccionarla aparecen tres nuevas pestañas: una variable, valores y fórmula. Para iniciar la captura de datos de esta VMR, se selecciona el tabulador “una variable”.

Figura 98. Datos generales de una VMR

Como es obvio, al inicio la variable no tiene datos de ningún tipo. En esta parte se le da un nombre (como aparecerá en todos los clientes que la quieran usar), unidades, el tipo de periodos, el periodo inicial de funcionamiento así como también una descripción y comentarios que se deseen. Los datos de submodelo y fórmula se generan hasta que se ejecuta el cálculo de niveles y submodelos, como ocurre en el modelo de un cliente.

Cuando una VMR no está destinada a tener fórmula, sus valores pueden obtenerse de dos formas: mediante captura manual o por medio de un agente. En la interfaz de la Figura 98 se puede seleccionar “SÍ” en “tiene agente”, para lo cual se activa el cuadro de datos para el agente.

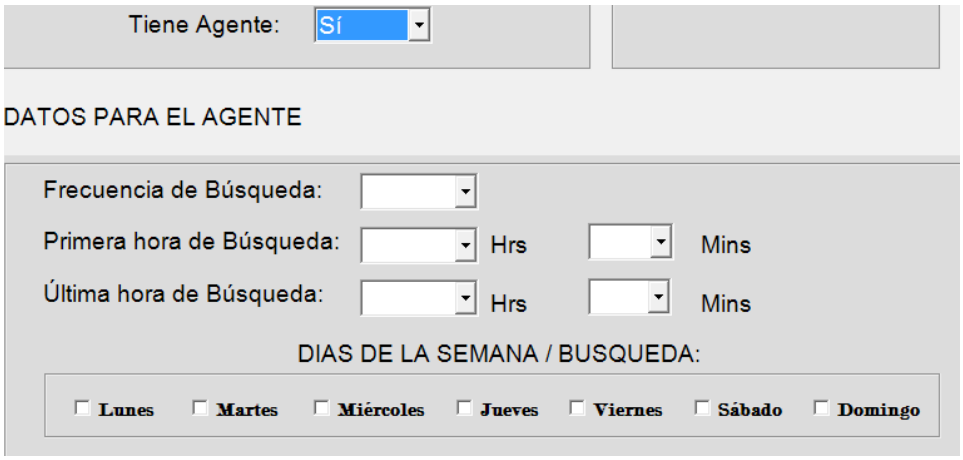


Figura 99. Datos de una VMR para el agente

Los datos principales son la frecuencia de búsqueda que en sí es la mínima periodicidad solicitada por los clientes, de dicha VMR. Está claro que al crearse no se tiene este dato ya que ningún cliente la ha solicitado, por esto se le asigna un valor default y se va ajustando conforme los clientes hagan sus solicitudes de periodicidad. También se debe asignar el horario y días de la semana en que esta VMR obtendrá actualizaciones de sus valores.

4.2.2. Formulas Entre VMR

En caso de que se indique que la variable creada será de tipo *calculada*, no puede tener agente ni valores de forma manual, sino que se calculan en base a otras VMR de abajo.

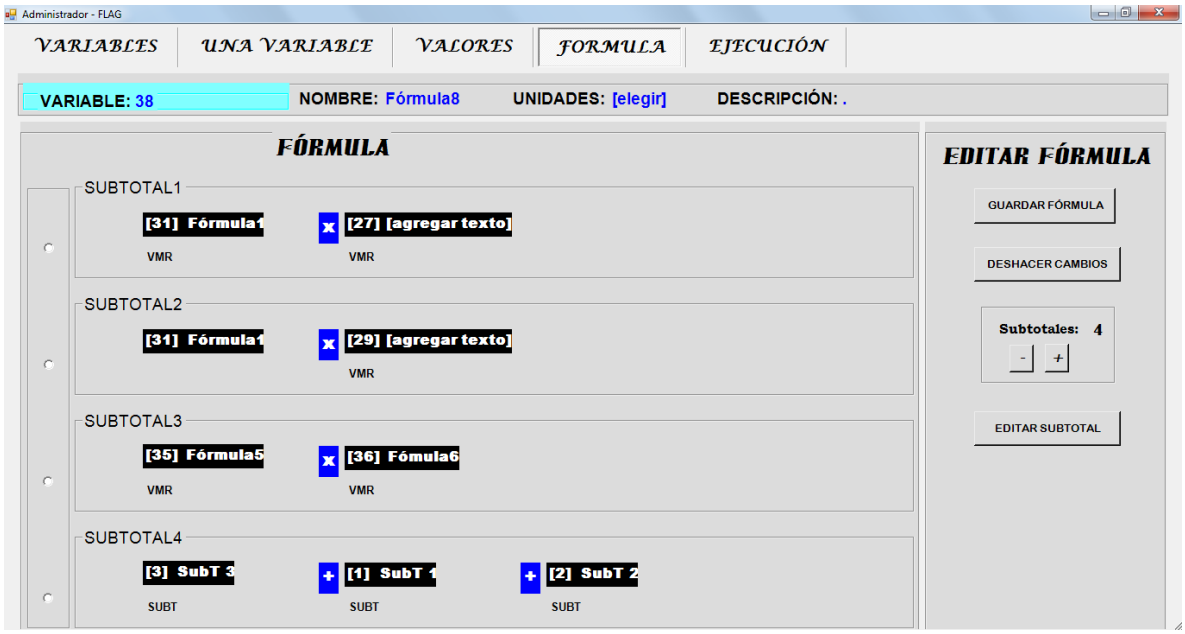


Figura 100. Ejemplo de una fórmula para una VMR

La captura de una fórmula, en el modelo de FLAG, es de la misma forma que se explicó para un modelo de un cliente (ver Figura 100); el uso de operandos, subtotales, operaciones y demás herramientas es el mismo. La única diferencia es que sólo ofrecerá dos tipos de operando: subtotal o una VMR.

4.2.3. Valores de Variables del Mundo Real

Los valores de las variables del mundo real se obtienen de tres maneras posibles: captura manual, agente o se calculan. Para el caso de que la VMR dependa de un agente, lo único es determinar los valores expuestos en la Figura 99 y asignarle un agente para que haga la búsqueda en la red, esto último se detallará más adelante. Los valores que se obtienen por el cálculo de una fórmula son fijos hasta que alguna que la integra como operando cambie por un agente o de forma manual, según sea la caso. Con esto queda claro que la periodicidad de una VMR calculada no se usa en ningún proceso. La captura manual de valores es también semejante a la que se hace con variables de cliente.

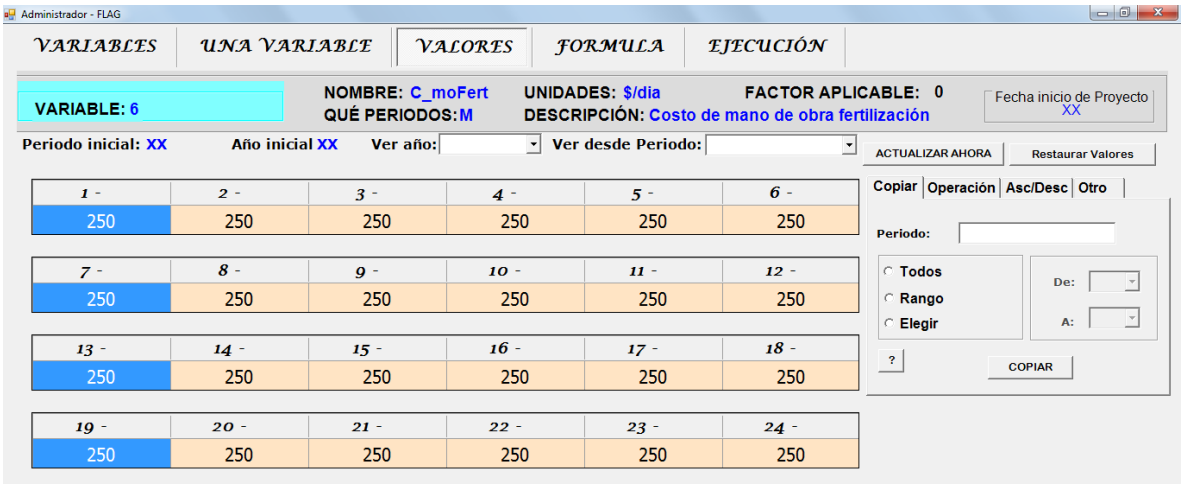


Figura 101. Captura y consulta de valores de una VMR

Las variables del mundo real también son vectores de 24 periodos, definidos por el usuario que la da de alta. En la Figura 101 se puede ver la herramienta de captura en la parte derecha, semejante a la descrita para variables del cliente; se puede capturar periodo por periodo, todos a la vez, un rango o solo los periodos que se elijan. Esta interfaz muestra los valores más actuales de la VMR elegida, no importa si es calculada, de abajo o si tiene agente.

4.3. Preparación del Archivo de Formulas del Mundo Real

Ya se explicó la captura de VMR en el modelo de FLAG así como los valores y la creación de fórmulas y se pudo constatar que dichos procesos son muy semejantes a los usados en el modelo de un cliente. La preparación del archivo de fórmulas en el modelo de FLAG no es la excepción.

El archivo de fórmulas en el mundo real consta de tres partes: registro base, lista ordenada de variables con fórmula y el arreglo de las fórmulas a calcular. La diferencia, con respecto al archivo de fórmulas de un cliente, es que aquí no se incluye información de criterios de alarmas ni submodelos. En primera, las alarmas están únicamente definidas para el cliente, a FLAG no le importa cuándo ni cuánto cambien las VMR.

Se decidió, también, no incluir la lista de submodelos (o lista de fórmulas a ejecutar cuando una VMR cambia) ya que, en general, el modelo de FLAG no será complicado, esto es, el nivel máximo de las fórmulas siempre será relativamente bajo.

Siempre que se capturen o modifiquen fórmulas, se vuelve a generar el archivo debido a que la composición y estructura de las fórmulas cambia y es necesario

crear nuevamente submodelos y generar niveles de forma correcta. El archivo base se actualiza y la lista de variables ordenadas también.

Cuando se hayan introducido variables y fórmulas al modelo de FLAG, se procede a ejecutar el cálculo de niveles y submodelos; es decir, siempre que haya algún cambio en el modelo, este módulo se ejecutará. Lo primero es extraer las variables de la base de datos y almacenarlas en estructuras temporales.

Tabla 39. Estructura temporal de las VMR

CAMPO	TIPO	DESCRIPCIÓN
Existe esta variable	Booleano	Verificación del uso de esta VMR
Tiene fórmula	Booleano	Verdadero si se ha definido fórmula para esta VMR
Es de arriba	Booleano	Verdadero si no pertenece a ninguna otra fórmula
Nivel	Entero	Posición jerárquica en el modelo
Pasada que actualizó nivel	Entero	Con este dato se verifica si aparece un loop (redundancia cíclica) en el modelo
Posición en el archivo	Entero	Número de orden en que se ejecutará su fórmula
De cuántas fórmulas es operando	Entero	Cantidad de fórmulas en las que aparece
Dad	Entero	Variable de cuya fórmula forma parte como operando
Submodelo	Texto	Numeración que define la posición de esta variable en el modelo de FLAG

Las variables, ya sean escalares o aleatorias, se almacenan, cada una, en un arreglo del tipo mostrado en la Tabla 39. El índice del arreglo denota el número de la variable que se procesa.

Acto seguido se procesa cada fórmula para obtener los niveles de cada variable en el modelo. Se recorre cada operando obteniendo el de mayor nivel, después se asigna este número de nivel más uno a la variable de la fórmula. En la variable temporal se guardan los valores del nivel, la fórmula que lo contiene y si dicha variable tiene fórmula. La “pasada que actualizó el nivel” es una variable de control que únicamente se usa en este proceso para la detección de las variables involucradas en algún bucle, si hubiera tales bucles.

Ya con los niveles calculados se generan los sinónimos para las variables que actúan como operando en más de una fórmula. Se recorren los operandos, uno por uno, primero para fórmulas de variables escalares, después de aleatorias. Se recorre cada operando, para cada uno se acumula la cantidad de veces que aparece esa variable en esa y otras fórmulas; el valor se guarda en la variable temporal en “de cuántas fórmulas es operando”. Cuando aparece en más de una fórmula se invoca la rutina “crea nuevo sinónimo” con los datos de la variable. El tipo de operando cambia a 2, que significa que en ese operando hay un sinónimo y no una variable; este campo, al iniciar el proceso, vale 1 por default para todas las VMR. Con esto las fórmulas tienen identificado cada uno de sus operandos como variable original, sinónimo o subtotal.

La creación de submodelos se hace recorriendo las variables de arriba, es decir, las que no pertenecen a ninguna otra fórmula y, por consecuencia, su dad vale 0. De cada una de ellas se desprende un submodelo tipo árbol con nodos y subnodos donde variables de abajo y calculadas (con fórmula) aparecen en hojas correspondientes al nivel que les fue asignado. A cada variable de arriba se le asigna un número entero consecutivo como submodelo, es decir, 1, 2, 3,… *n*. Siendo *n* el número total de variables de arriba. Si la variable de arriba tiene fórmula, se procesan sus operandos y a cada uno se le asigna un submodelo con un número entero consecutivo. Por ejemplo, se está procesando la variable calculada de submodelo 3, si tiene 5 operandos sus submodelos serán 31, 32, 33, 34 y 35; es decir, del número uno al cinco precedido por el submodelo que ya tiene la fórmula que contiene esos operandos. Después de asignar submodelos a los operandos se procesa cada uno de ellos determinando si sus variables tienen fórmula, de ser así se procede a hacer lo mismo hecho previamente. Por ejemplo, si la variable que tiene el submodelo 34 tiene fórmula y ésta tiene tres operandos, los submodelos de cada uno de ellos serían 341, 342 y 343. Este proceso se hace mediante una rutina recursiva que termina hasta que todos los operandos son variables de abajo. Al terminar este proceso se actualiza el valor de “submodelo” en la variable temporal de VMR.

Se almacena en el archivo plano de fórmulas y en la base de datos la información obtenida en estos tres procesos. De cada variable, en la base se guarda si es de arriba, su nivel, dad y submodelo. Para el archivo, se vuelve a generar la lista ordenada por nivel en orden ascendente así como la posición que le toca en el archivo. Cada fórmula se guarda después de la lista ordenada, una por una de la misma forma que aparece en la lista. Cuando se invoque la ejecución del modelo FLAG, se leerá este archivo y sus fórmulas se ejecutarán en forma ordenada como están guardadas.

4.4. Preparación del Archivo de Criterios de Ejecución de Los Modelos de Los Clientes

4.4.1. Introducción

Los criterios de ejecución se determinan sobre las variables del mundo real que el cliente está usando en su modelo. Esto se explicó en el capítulo dedicado a las funciones que tiene cada cliente para actualizar su modelo, sus fórmulas y los criterios de alarma y de ejecución.

Cada cliente introduce estos criterios a su base de datos. En el caso de los criterios de ejecución, el FLAG concentra los criterios de todos los clientes en una tabla de su base de datos (la del servicio). Posteriormente se copian estos criterios

a un archivo plano para que el “motor” del FLAG no tenga que procesar la tabla de la base de datos: sólo se carga el archivo plano a memoria, puesto que – como se verá y se planificó vía el diseño de las estructuras – tienen la misma estructura que los arreglos del programa que los usan para establecer cuáles modelos deben ejecutarse cuando los agentes de búsqueda proporcionen valores actualizados.

Es importante tener en cuenta esto porque si una VMR en el modelo del cliente no tiene criterios, no necesariamente estará actualizada conforme a los datos de FLAG o por lo menos no en todo momento. La única forma en que el total de variables de un modelo se actualicen es cuando se ejecutan todas las fórmulas. Se puede decir que un modelo se actualiza cada vez que lo hace la VMR de mayor frecuencia. Estrictamente no es así, pero si la periodicidad de una variable se especifica como baja, es porque cambia en intervalos de tiempo muy largos y no es necesario actualizarla ni ejecutar el submodelo correspondiente.

Los criterios de las variables del mundo real tienen cuatro componentes: el número de la VMR, la regla que se aplica, un límite inferior y un límite superior. Estos límites definen un intervalo alrededor del valor de una variable: si el valor nuevo cae fuera de dicho intervalo, se invoca la ejecución del modelo del cliente (o de un submodelo, si procede esta alternativa). La reglas, mostradas en la Tabla 40, se refieren al tipo de cambio en los valores que se evaluará, es decir, porcentaje o cantidad escalar; también define el valor de comparación que puede ser el último que actualizó el agente en FLAG o el último que provocó una ejecución en el modelo del cliente.

Tabla 40. Reglas para criterios de ejecución

REGLA	COMPARAR CON	EL CAMBIO ES
1	Agente	Escalar
2	Valor mío	Escalar
3	Agente	Porcentual
4	Valor mío	Porcentual

Tabla 41. Criterios para varias VMR en un modelo de un cliente

Número de variable	Regla	Límite Inferior	Límite Superior
2000001	1	1	1
2000002	3	5	5
2000003	1	120	120
2000004	1	50	60
2000005	2	38	42
2000006	2	0.5	0.1
2000008	4	1	2.5
2000009	3	20	20
2000010	1	5	10

La Tabla 41 muestra un ejemplo típico de los criterios capturados para 10 variables del mundo real. Siempre, al inicio de cada jornada de trabajo del servicio FLAG, la información de criterios debe estar al día para que la evaluación de los cambios sean los correctos. Antes de generar el archivo de criterios de todos los clientes se deben extraer los criterios de cada uno a una tabla común en FLAG denominada criterios_VMR_clientes. En la Tabla 42 se muestra su estructura.

Tabla 42. Campos de la tabla criterios_VMR_clientes

CAMPO	TIPO	DESCRIPCIÓN
Número de cliente	Entero	Largo
Número de VMR	Entero	Largo
Número de VMR en el modelo del cliente	Entero	Largo
Ejecutar si cambia esta VMR aleatoria	Booleano	Cuando la VMR usada es aleatoria, este campo se selecciona para ejecutar todo el modelo si cambia aunque sea un poco
Regla	Byte	Se selecciona una de cuatro posibles
Periodicidad	Entero	El tiempo mínimo con el que esta VMR se actualizará y evaluará el criterio
Límite inferior	Flotante	
Límite superior	Flotante	

Los campos llave son los dos primeros (cliente y variable). Para cada cliente hay un determinado número de VMR que está usando y para cada una de ellas existe un criterio previamente definido. Lo que sigue es leer esta tabla y juntar la información en un solo archivo de criterios.

4.4.2. Preparación del archivo

Para la preparación del archivo de criterios de ejecución se debe leer la tabla criterios_VMR_clientes y guardar los criterios de forma compacta en estructuras de datos específicas, de modo que su lectura sea fácil y rápida.

El archivo se compone de cuatro secciones:

- Registro base
- Lista de modelos
- Lista de VMR
- Arreglo de criterios por variable.

Las estructuras usadas se explican a continuación.

Tabla 43. Registro base de criterios

CAMPO	TIPO	DESCRIPCIÓN
Cuántas VMR	Entero	Total de variables en FLAG
Cuántos clientes	Entero	Total de modelos/clientes que usan el servicio
Ubicación de la primer variable con criterios	Entero	Byte de la posición en el archivo de criterios de los datos de la primer variable
Actualización pendiente	Boolean	Este campo es verdadero cuando los criterios, en la base de los clientes, han sido modificados

El registro base es la parte inicial del archivo; tiene información sobre las secciones que lo componen (ver Tabla 43). Permite obtener los tamaños de los arreglos de clientes y variables, así como el byte (posición relativa en el archivo) donde comienzan los criterios. Cuando no hay actualización pendiente, no se ejecuta la preparación puesto que los criterios no han sido modificados para ningún cliente.

La posición de la primera variable con criterios está inmediatamente después de las tres secciones anteriores, es decir, del registro base, de la lista de variables y la lista de modelos. El número de variables se obtiene de la tabla de VMR en FLAG y la de modelos de la tabla “Accounts” de la base (ver Tabla 45). Con los tamaños de cada lista y el tamaño de cada estructura se obtiene el peso del archivo en bytes de esas tres secciones, por lo que la posición de la primer variable con criterios es el peso del archivo, al momento, más un byte.

La lista de clientes es la segunda sección del archivo de criterios. Es un arreglo de estructuras; los campos de la estructura se muestran en la Tabla 44.

Tabla 44. Estructura “lista de clientes”

CAMPO	TIPO	DESCRIPCIÓN
Número de cliente	Entero	
Cuántas VMR	Entero	Las VMR que actualmente usa este cliente en su modelo
Directorio	Texto	Ubicación física de sus archivos y bases de datos en el servidor FLAG

Para obtener los datos de los clientes, así como de las variables del mundo real, es necesario leer la base de datos de FLAG. La información general de cada cliente que está adscrito al servicio se almacena en una tabla con nombre “Acounts”. Su estructura se muestra a continuación.

Tabla 45. Tabla *Accounts* o miembros del servicio FLAG

CAMPO	TIPO	DESCRIPCIÓN
Número de modelo	Entero	
ID de cliente	Entero	Número que identifica a cada uno de los clientes del servicio

Directorio de trabajo	Texto	Ubicación física de sus archivos y bases de datos en el servidor FLAG
Cuántas VMR	Entero	Cantidad de VMR que actualmente usa este cliente en su modelo
Proceso pendiente	Boolean	Si hay modificaciones al modelo de un cliente y la información en FLAG aún no ha sido actualizada, este campo es verdadero.
Fecha de alta	Fecha - Hora	
Última ejecución	Fecha - Hora	

El arreglo de clientes de la estructura mostrada en la Tabla 44 se llena con la información de la Tabla 45 de *Accounts* en la base. Se recorre cada registro y se guarda, en el arreglo, de forma consecutiva cada modelo existente con sus datos.

La siguiente sección del archivo contiene las variables del mundo real: se trata de un arreglo con la estructura de la Tabla 46.

Tabla 46. Lista de variables del mundo real

CAMPO	TIPO	DESCRIPCIÓN
Número de variable	Entero	
Cuántos modelos	Entero	Cantidad de modelos que usan esta variable
Ubicación byte	Entero	Posición de los criterios de esta variable en el archivo (sección 4)

Para obtener la información de este arreglo se leen dos tablas de la base de FLAG: primero la de VMR y después la Tabla 42 de criterios. El tamaño del arreglo es la cantidad de VMR “de abajo” – es decir, que no tienen fórmula - definidas en el sistema, por lo que se recorre la tabla de VMR y se anexa cada una al arreglo de la Tabla 46. Para obtener la cantidad de modelos por VMR se lee la Tabla 42 filtrando por la VMR correspondiente. La cantidad de modelos resultante se almacena en el arreglo.

Para obtener la ubicación de los criterios de cada variable, se acumula el número de modelos que usa en una variable temporal. Cuando se lee la siguiente VMR, la ubicación de sus criterios es el número acumulado de modelos multiplicado por el tamaño de cada uno (41 bytes). Si la primera variable tenía 8 modelos, la siguiente comienza en el byte número 41 por 8 más la posición de la primera. Si la segunda tiene 4 modelos, la tercera inicia en la posición 12 (es decir, 8 + 4) por 41 sumado a la primera. Este proceso termina cuando se leen todas las VMR del sistema.

En este punto ya se generaron tres de las cuatro secciones que componen el archivo sólo resta obtener la parte más importante, los criterios por cada cliente.

En la Tabla 47 se muestra la estructura de cada elemento del arreglo que compone la última sección del archivo.

Tabla 47. Criterios de cliente por variable

CAMPO	TIPO	DESCRIPCIÓN
Cliente()	Cliente_criterio	Arreglo de criterios de clientes

El tamaño del arreglo es el número de VMR definidas; sin embargo, el tamaño de cada elemento es variable. Esto es porque, como se mencionó antes, al momento de generar la lista de VMR, se obtiene el número de modelos que la usan, y varía en cada una. La Tabla 48 detalla la información que compone a cada elemento del arreglo "cliente" que, a su vez, forma parte del arreglo de VMR de la sección cuatro del archivo de criterios.

Tabla 48. Criterio de un cliente

CAMPO	TIPO	DESCRIPCIÓN
Número de cliente	Entero	
Número de VMR en el modelo del cliente	Entero	Número con el que el cliente conoce esta VMR.
Valor a comparar	Flotante	Es el último valor de ésta VMR con el que el modelo de este cliente fue ejecutado.
Ejecutar si es aleatoria	Boolean	Este campo es verdadero si esta VMR es aleatoria y si el cliente desea que su modelo sea ejecutado cuando ésta cambie.
Regla	Byte	Define el tipo de valor a comparar y la forma en que se evalúa el cambio.
Periodicidad	Entero	Tiempo mínimo para ejecutar la evaluación del criterio de esta VMR para este cliente. Las unidades se manejan en minutos.
Límite inferior	Flotante	
Límite superior	Flotante	
Última actualización	Fecha – Hora	Último cambio que hubo en este criterio.

El proceso se hace de la siguiente forma:

- Se redimensiona el arreglo "criterios de clientes por variable" al total de VMR que hay en FLAG.
- Se lee cada una de las VMR del arreglo previamente generado (Tabla 46).
- Se lee la cantidad de modelos que la usan.
- Se redimensiona cada sub arreglo "cliente" al número de modelos obtenido en el paso anterior.
- Se filtra la tabla "clientes – criterios" (Tabla 42) con el número de la VMR que está en proceso.
- Los registros filtrados se ordenan por el número de cliente.
- Se lee cada cliente de la tabla ya filtrada y ordenada.

- De esta forma se obtiene el criterio de un cliente y se almacena en el arreglo “criterios de clientes por variable” en la posición de la VMR que le corresponde e igualmente en el “cliente” que le toca.
- Los datos extraídos son los que se muestran en la Tabla 48.
- Al recorrer todos los “clientes” de una VMR se remueve el filtro y se continúa con la siguiente (VMR).
- Una vez terminado el recorrido de las VMR, se almacena este arreglo en el archivo de criterios, en la posición de la “primer variable con criterios”.

Más adelante, en la “invocación de modelos”, este archivo se leerá y se copiará a memoria en clases con estructuras un poco diferentes a las manejadas aquí, para que el proceso de aplicación de criterios ante cambios de valores de las VMR sea lo más eficiente posible.

5. EL MOTOR DEL FLAG

5.1. Agentes

5.1.1. Introducción

En FLAG las VMR son por definición variables dinámicas ya que se trata de variables de entorno sujetas a cambios virtualmente no predecibles, ajenas a los clientes y al servicio FLAG. En el capítulo 4 se menciona el uso de las VMR en el servicio, así como la creación de las mismas, el modelo que pueden conformar y la incorporación de sus atributos. Entre estos últimos están los agentes que se encargarán de sus valores. Se explica la manera de asignarlos a cada VMR y la información que necesita el agente; sin embargo no se da mucho detalle de los agentes en sí, eso es parte de lo que se detallará en la actual sección.

El tipo de agente usado en FLAG es el llamado agente de información. Este concepto es muy popular en los procesos que se realizan a diario en internet, por ejemplo, en los motores de búsqueda, en consultas académicas (o no) de bases de datos extensas, investigación de mercado, etc. Las características y ventajas de los agentes hicieron que en el diseño de FLAG se contemplaran como el recurso de obtención de valores de las variables del mundo real. Fue indispensable determinar el modo de obtención de dichos valores ya que el servicio FLAG está pensado para ejecutarse en tiempo real, es decir, usando siempre los valores más actuales de las VMR.

El tema de los agentes fue inicialmente concebido por Bauer (2008) en la primera versión del servicio FLAG. Posteriormente Díaz (2009) desarrolló un programa en el que se obtienen valores de variables a partir de agentes informáticos que son lanzados a la web. Recientemente, este concepto fue actualizado y mejorado por Varela (2014) que usa estructuras (*ad hoc*) muy diferentes a las usadas por Díaz (2009) y optimizando exponencialmente el tiempo de los procesos incorporando ejecución multi–hilos.

El servicio FLAG se ha ido desarrollando en forma modular, es decir, por partes a lo largo de varios años y siempre supervisados por el Dr.Bauer. En lo que concierne a la parte de agentes, lo último y más reciente realizado está en la investigación de Varela (2014), cuyas estructuras, bases de datos y procesos fueron diseñados en conjunto con la actual investigación. A continuación se describe el programa que crea los agentes y la forma en que son ejecutados.

5.1.2. Preparación del programa

Básicamente el proceso consta en lanzar los agentes mediante dos elementos fundamentales: la página web de la cual se obtiene la información correspondiente a una variable y el método mediante el cual se extrae el valor de interés de dicha página (Varela, 2014).

5.1.2.1. Tabla Agentes

Los agentes deben contar con información previa antes de ser lanzados a la web, para esto se diseñaron tablas que serán usadas por los agentes para que éstos las lean y de esa manera puedan completar su objetivo: conseguir los valores actuales del mundo real.

Tabla 49. Campos y características que se almacenan en la tabla de AGENTES

CAMPO	TIPO	DESCRIPCIÓN
Agente	Entero	Número consecutivo del agente
Nombre	Texto	Nombre con que se conocerá al agente
URL	Texto	Fuente de información
Frecuencia de búsqueda	Entero	Periodicidad mínima de búsqueda de información
Primera hora de búsqueda	Texto	Hora del día para la primera consecución del valor
Última hora de búsqueda	Texto	Hora del día para la última consecución del valor
Días de búsqueda	Texto	Días de la semana en que se buscará el valor
Cuántas variables	Entero	Cantidad de variables asociadas al agente

Los datos que se muestran en la Tabla 49 dependen de los que hayan sido definidos para las variables a las cuales está asociado el agente. En la sección de ABC de las VMR se puede ver que para cada variable deben definirse los datos de la periodicidad, horario y días de la semana en que debe obtener sus valores. Cuando se asigna una variable a un determinado agente, lo que hace este último es tomar esta información y almacenarla en su correspondiente registro de la Tabla 49. De esta manera el agente sabe la frecuencia y horario de búsqueda.

5.1.2.2. Abc de agentes

Para ilustrar el manejo de los agentes se creará uno nuevo a manera de ejemplo, el cual es el precio de las acciones de Facebook. En la Figura 102 se puede ver la interfaz del programa de altas, bajas y cambios de agentes; también se ve que ya hay creados tres y se agregará uno más.

Figura 102. Interfaz inicial de agentes de búsqueda

Después de seleccionar “agregar” en el programa, aparece un cuadro con el número de agente y sus datos iniciales. En la Figura 103 se ve el ejemplo, un nuevo agente denominado “Acciones FCBK” más la página web donde se podrá encontrar la información

Figura 103. Inicialización de un nuevo agente

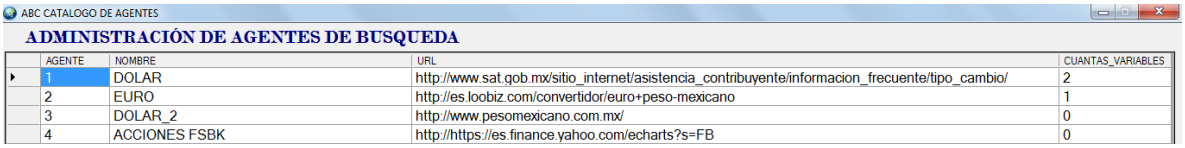


Figura 104. Nuevo agente agregado a la lista.

Cuando se inicializa un nuevo agente no tiene datos, solamente la dirección web que debe “parsear”, lo siguiente es la forma en que éste será usado, esto es, asignar variables al agente recién creado. En la Figura 104 se ve que aún no hay VMR asignadas, entonces lo que sigue es asociarle una variable, como se ve en la Figura 105.

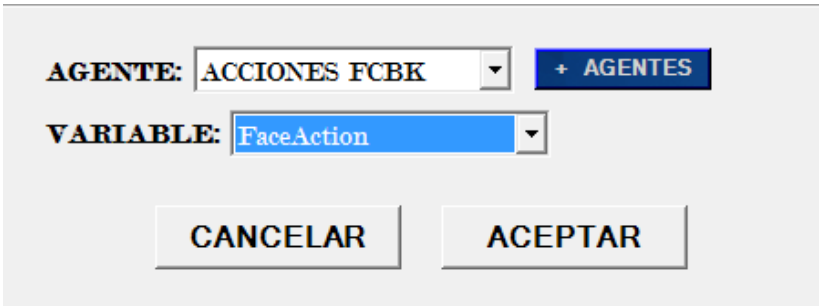


Figura 105. Asignación de agente a VMR

Previamente se creó una variable que almacenará los datos de las acciones de Facebook denominada “FaceAction”, a ésta se le asignó el agente recientemente creado (ver Figura 105). A continuación, en la página web parseada, como se ve en la Figura 106, se proporcionan datos de una palabra y la cantidad de caracteres que se guardarán para que el agente obtenga los valores correctos.

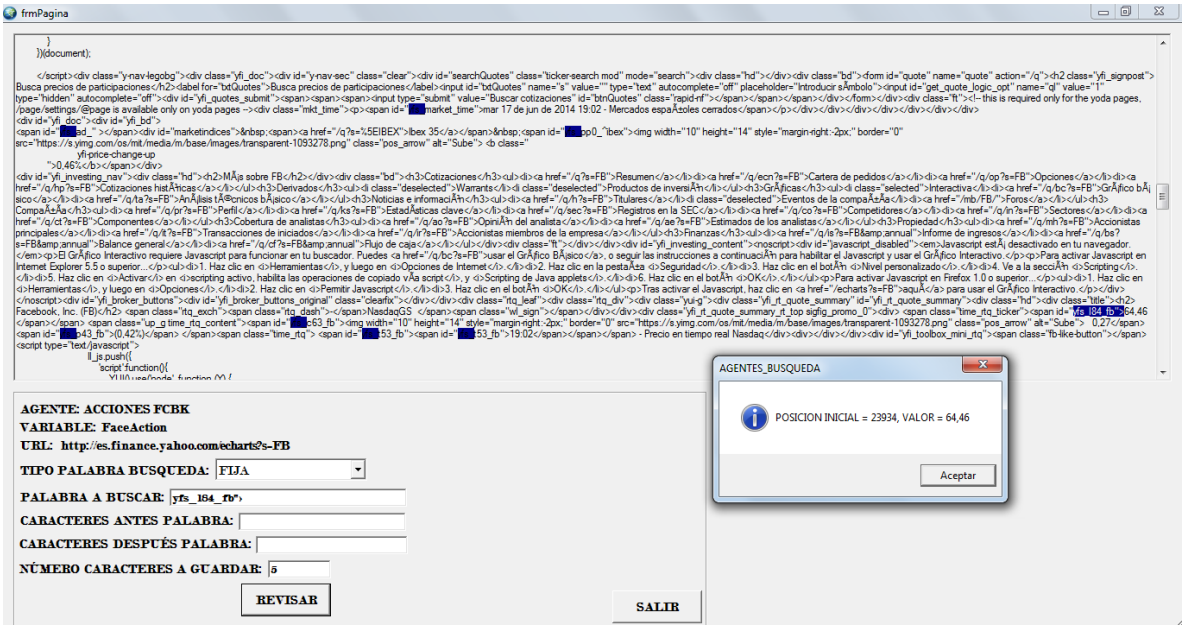


Figura 106. Parseo de la página web para encontrar el valor deseado

De esta manera se ha creado un nuevo agente que se encargará de obtener los datos del precio de las acciones de Facebook. Lo que sigue es entender el proceso concurrente que ejecuta los envíos.

5.1.3. Envío de agentes (multi-hilo)

Los agentes tienen diferente periodicidad, sin embargo, según lo definido en el sistema, las periodicidades varían entre 1 minuto, 10 y 60, que son los horarios con que cada agente se impondrá a buscar valores de las VMR que le corresponden.

Las propiedades de los agentes, como se dijo antes, se guardan en una tabla en la base de datos de FLAG, sin embargo, en la ejecución del proceso toda la información se manda a memoria mediante unas estructuras especiales (Varela, 2014). Esto es porque se debe ahorrar lo más posible el tiempo en ejecución, tomando en cuenta que el servicio FLAG hará uso de miles de variables que, a su vez, utilizarán miles de clientes en sus modelos. El tiempo de ejecución es crucial, lo mismo se verá en el apartado de “Evaluación de criterios de ejecución”, por eso es poco viable leer una base de datos en un proceso de este tipo.

Computacionalmente hablando, un hilo crea y controla un subproceso, establece su prioridad y obtiene su estado (Microsoft, 2014). Esto quiere decir que el programa que se encarga de obtener el valor nuevo de una variable del mundo real, lo hace al mismo tiempo que los demás agentes, es decir, cada uno se vuelve un subproceso y se controla al mismo tiempo que los demás.

El proceso de envío de agentes es de la siguiente forma:

A) Inicia variables de la base de datos. Carga las estructuras que se quedarán en memoria con los datos de los agentes y las variables que les corresponden.

B) Manda agentes. Se crea un ciclo con una hora inicial y una final. A lo largo de este tiempo se estarán enviando los agentes según la periodicidad que tengan definida. Al pasar un minuto se envían todos los que tienen frecuencia mínima de un minuto, después de 10 se hará lo mismo y al término de una hora lo propio para los correspondientes.

Cabe decir que al pasar 10 minutos se envían los que tengan tal periodicidad pero también los de un minuto. Y en el transcurso de una hora se envían los que tengan periodicidad una hora y menor.

C) Esperar la llegada de todos los agentes. Cuando ocurre se envía una tanda de agentes el proceso debe esperar a que todos tengan una respuesta, es decir, que vuelvan con los valores encontrados o, si hay un detalle o problema con la fuente de información, reportarlo como error; pero siempre debe haber una respuesta.

D) Crear el archivo de valores. Cuando los agentes de una sesión de envío han vuelto, ya cuentan con el valor nuevo en su estructura. Lo siguiente es leer cada uno de ellos y guardar en una variable temporal compuesta la VMR y su valor. La variable es una estructura de dos campos: VMR de tipo entero largo y Valor de tipo punto flotante. Esta nueva variable se almacena en un archivo denominado “ValoresVMR.Flag”. El proceso es de la siguiente manera:

- Se lee la carpeta donde se guardan los archivos de trabajo en el servidor FLAG, para buscar el archivo antes mencionado.

- Si el archivo se encuentra quiere decir que los valores anteriores aún no han sido procesados por la *evaluación de criterios de ejecución*. En este caso se sobre escribe sobre el ya existente. En caso contrario se crea uno nuevo.
- Al inicio del archivo se deja el espacio de 4 bytes para almacenar el número total de variables que cambiaron. A partir de ahí se guarda la información.
- Se lee cada agente procesado y se extraen los datos de la variable y valor obtenido en la estructura antes mencionada.
- Se introduce el par de datos, uno en seguida del anterior en el archivo
- Al terminar de leer los agentes, se guarda en la primera posición el total de variables que cambiaron
- Se cierra el archivo.

Los agentes de información son la parte que hace que el servicio sea dinámico y actual. Sin ellos la consecución de valores sería extremadamente más complicado, solamente pensar el hacerlo de forma manual, ya se torna tedioso, además de la mano de obra que se ocuparía. En cuanto al funcionamiento, se podrían diseñar programas más eficientes para el envío y recuperación de información, pero el hecho de utilizar ejecución multi-hilos ya es bastante ventaja, es decir, el tiempo que tarda la recuperación de datos de todos los agentes es el mismo que el del más tardado.

Por ahora la consecución de valores de las VMR se ha definido de la forma que se explica en esta sección, sin embargo y como en todos los sistemas, esto es mejorable; por el momento es el método más eficiente que se usó, de ser necesario, en un futuro se puede modificar para bien del servicio.

5.2. Evaluación de Criterios de Ejecución

5.2.1. Introducción

La otra parte que compone el motor del FLAG es la evaluación de los criterios de ejecución de los clientes para decidir cuáles modelos serán ejecutados. Para esto el programa debe realizar algunos procesos necesarios para evaluar las variables que traen los agentes.

El sistema FLAG tiene características especiales en cuanto al proceso de ejecución. Cuando está ya en funcionamiento, la invocación de modelos se ejecuta con la periodicidad mínima que tienen los agentes, es decir, cada minuto, que es cuando se obtienen valores de las VMR que cambian. La cantidad de ellas que llega depende de la periodicidad que se haya definido en cada una. Cada minuto del día el proceso de invocación se está ejecutando, con una cantidad

diferente de ellas cada vez. El programa tiene una pausa al inicio del día, que puede ser definida por el usuario administrador, donde se actualiza la información que usa el sistema, como los son las bases de datos de los clientes, sus archivos de fórmulas y valores, la ejecución de niveles y submodelos del modelo FLAG, actualización de la tabla de criterios de ejecución de todos los clientes, la creación del archivo de criterios y cualquier modificación a VMR, agentes o, incluso, altas y bajas de estas dos. Esto es para que los resultados de la ejecución sean congruentes con la información que se está usando.

Las variables del mundo real están definidas en el modelo de FLAG, cuando cambia alguna de ellas, si forma parte de una fórmula, se debe ejecutar el modelo y después sumarse a la lista de las que cambiaron. Con esto, la lista de VMR a procesar crece. Las variables de FLAG están siendo usadas por muchos clientes al mismo tiempo, es por eso que este programa identifica cuáles modelos son afectados y cuáles deben ser ejecutados cada vez.

5.2.2. Preparación del Programa

El proceso que invoca la ejecución de modelos hace uso de dos archivos: el de VMR que cambiaron y el de criterios de ejecución. Éstos se encuentran en una carpeta fija de FLAG, para iniciar la invocación se leen ambos archivos y se mandan a memoria.

Como se explicó en la parte del cliente, las VMR están sujetas a cambios constantes que se ven reflejados en su modelo con la periodicidad que él mismo definió. A parte de esto, también define criterios para cada una, es decir, límites en sus valores en los que si el cambio los sobrepasa, el submodelo debe ser ejecutado. Los criterios están en la base del cliente que a su vez es exportada a FLAG, aquí el sistema lee cada una de ellas y llena la tabla de *clientes_criterios* donde se concentra toda la información. Lo siguiente es la creación del archivo de criterios, que como se explicó en el apartado de preparación del mismo, lee la tabla y guarda la información en un archivo plano.

El archivo de criterios tiene una estructura definida específicamente para guardar la información de forma compacta, esto porque cuando la cantidad de usuarios es elevada, el archivo será, casi siempre, muy pesado y es necesario utilizar el espacio en disco lo más eficiente posible.

En el archivo está la información de todos los clientes que están adscritos a FLAG mediante un arreglo con la estructura mostrada en la Tabla 50.

Tabla 50. Estructura de clientes en FLAG

CAMPO	TIPO	DESCRIPCIÓN
Número de cliente	Entero	Es el número con el que fue registrado este cliente
Cuantas VMR	Entero	El número de variables del mundo real que está usando
Directorio	Texto	Es la dirección de la carpeta en FLAG donde se encuentran los archivos de este cliente.

Tabla 51. Estructura de la clase que almacena la información de un cliente

CAMPO	TIPO	DESCRIPCIÓN
Número de la variable a ejecutar	Entero	Al evaluar los criterios, si solamente cambió una variable, ésta se almacena aquí, para después ser ejecutado su submodelo
Cambió más de una VMR	Booleano	En el proceso de evaluación de criterios se determina cuando más de una VMR en el modelo de este cliente ha cambiado
Cuantas VMR	Entero	El número de variables del mundo real que está usando
Directorio	Texto	Ubicación de los archivos de este cliente
Ultimo cliente	Entero (shared)	Esta es una variable compartida de la clase, aquí se almacena el último cliente que ha de ser procesado en la evaluación.

El número de cliente, la cantidad de VMR que usa y el directorio que contiene su información se extraen en un arreglo de instancias de una clase con la estructura mostrada en la Tabla 51.

Cada elemento del arreglo se instancia con los números de cliente del arreglo *listaCliente* como índices; cabe decir que el arreglo no es necesariamente consecutivo. Habrá igual número de instancias como clientes. Cada instancia se inicializa con la variable a ejecutar igual a cero y “cambió más de una variable” como falso. La variable compartida *ultimoCliente* se usa para guardar el cliente con el número más alto. Con todo esto se puede acceder de forma directa a la información de cada cliente en memoria. De este arreglo de clases se obtiene la lista final de modelos junto con la variable del submodelo que debe ejecutarse; si llega con valor cero, se ejecuta todo el modelo.

Los criterios se guardan en la clase denominada “unaVMR” detallada en la Tabla 52. Al inicio del proceso se instancian las VMR que han sido usadas por al menos un cliente, la clase “unaVMR” las mantiene en memoria hasta el término del proceso, dentro de cada instancia está la información de cada cliente respecto a dicha VMR.

Tabla 52. Estructura de la clase que almacena y procesa las variables del mundo real al ser evaluados los criterios de ejecución.

Entes de la clase	CAMPO	TIPO	DESCRIPCIÓN
Composición de la estructura CLIENTES	Número de cliente	Entero	
	VMR en el modelo del cliente	Entero	Número de la VMR de la actual instancia de esta clase, en el modelo del cliente
	Valor a comparar	Flotante	Último valor de la VMR que actualizó el modelo de este cliente
	Regla	Byte	Definida por el cliente en su modelo para esta VMR
	Límite inferior	Flotante	Valor mínimo permitido para invocar ejecución
	Límite superior	Flotante	Valor máximo permitido para invocar ejecución
	Última actualización	Date	Fecha de la última actualización de la VMR para este cliente
	Periodicidad	Byte	Definida por el cliente
Valor anterior		Flotante	Referente a la VMR que se está procesando
Valor nuevo		Flotante	Referente a la VMR que se está procesando
Cliente ( )		Clientes	Arreglo de la estructura tipo “clientes”. Aquí se almacena la información de todos los clientes que usan la VMR de la instancia de esta clase
Última actualización		Date	Fecha en que se renovó el valor de la VMR. Generalmente es cuando llegan desde los agentes

Al igual que con la clase clientes, ésta también se usa como arreglo en el que el índice de cada elemento es el número de variable. En cada uno se almacena el valor obtenido en la ocasión anterior por los agentes, el valor nuevo, la fecha y hora de la última actualización y un arreglo de clientes donde está el criterio que él definió para esa variable. Los valores anterior y nuevo están actualizándose constantemente mientras transcurre el proceso a excepción de la primera ejecución del día donde los obtiene del archivo de valores, que se encuentra en FLAG. Por su parte, los criterios de los clientes que usan las variables están en esta clase, se cargan desde la primera ejecución a partir del archivo de criterios decrito en el capítulo anterior. Estos datos no se modifican en el transcurso del día, sus datos no cambian, naturalmente con la excepción de “valor a comparar” y “última actualización“ que se usan en el proceso.

El hecho de usar clases para este proceso se debe a que el acceso es mucho más rápido que si se usaran estructuras. En éstas se desperdiciaría mucho espacio si el índice fuese el número de variable ya que puede ocurrir que muchas no se usen o estén dadas de baja. Si se usara un arreglo del tamaño de la cantidad de variables que estén vigentes, es decir, las que sí se van a usar, el problema sería encontrar el elemento que corresponde a tal variable porque el índice ya no correspondería a ningún criterio de uso. En ese caso se podría hacer una búsqueda binaria en la lista (los criterios se leen del archivo en orden ascendente de número de cliente). En cambio, usando clases se redimensiona el arreglo al

tamaño de la mayor variable, pero sólo se instancian las que se van a usar. Dimensionar un arreglo de instancias de clases no ocupa memoria (hasta que se intancien lo hacen).

Tabla 53. Estructura de la lista de VMR que cambian

CAMPO	TIPO
Número de variable	Entero
Valor	Flotante

El programa lee el archivo que fue creado por el programa de agentes. Hay una lista de variables con valores nuevos de la estructura mostrada en la Tabla 53 y lo carga a memoria como un arreglo. Si el archivo de agentes no existe quiere decir que en ese minuto no hubo cambio en variables por lo que se espera hasta el siguiente. Después de leerlo, se elimina el archivo.

Los valores de las VMR que llegan de los agentes se actualizan en su correspondiente archivo plano y antes de procesar la lista se debe ejecutar el modelo de FLAG. Esto no siempre será necesario ya que habrá muchas variables que no aparezcan en fórmulas. Sólo se ejecuta si hay por lo menos una variable de las que cambiaron de valor en alguna fórmula. La ejecución del modelo FLAG produce una lista de variables calculadas junto con el valor nuevo que obtienen; esta lista se anexa a la original de VMR que cambiaron.

5.2.3. Invoca Modelos de los Clientes

Lo siguiente es procesar la lista final de VMR que cambiaron (las de agentes más las calculadas). A continuación se explica cada paso que sigue la rutina para encontrar los modelos que se invocarán en la ejecución:

- Se recorre cada elemento de la lista. Los siguientes pasos se hacen para cada una de las VMR.
- Se actualizan valores en la clase *c_unaVMR*. El valor “anterior” se sustituye por el “nuevo” y el “nuevo” por el que llega en la lista.
- Se comparan estos dos valores, si son iguales se salta todo el proceso y continua con la siguiente VMR de la lista.
- Se recorren los elementos del arreglo cliente que está dentro de la clase ya mencionada. Los siguientes pasos se hacen para cada uno de ellos.
- Se obtiene la periodicidad actual. Para esto se calcula la diferencia entre la hora actual y la última en que se actualizó esa variable para ese cliente. Si el valor es mayor o igual que la periodicidad especificada por el cliente, sigue el proceso, de lo contrario continua con el siguiente cliente.

- Para VMR aleatorias: si el cliente definió ejecutar su modelo cuando su VMR aleatoria cambia, entonces en la instancia de la clase *c_unCliente* que corresponde al cliente que se está procesando, el valor de "cambió más de una variable" se actualiza como verdadero para que se ejecute todo el modelo. Comentario: en general no hay agentes para variables aleatorias; éstas se construyen a partir de otras con fórmulas del modelo. Pero el programa contempla esa posibilidad por si se modifican estas circunstancias.
- Para variables escalares se evalúan los criterios de ejecución del cliente de la siguiente forma:
    - Se calcula la diferencia entre el valor nuevo y el anterior. Si la regla usada por este cliente es 1 o 3, el valor anterior es el del agente; en cambio si usa la regla 2 o 4, el valor anterior es el corresponde al último usado del cliente, el almacenado en "valor a comparar".
    - Se calcula el porcentaje del cambio. Para esto se divide la diferencia entre el valor anterior y se multiplica por 100. Análogamente se calcula con su respectivo valor anterior.
    - Se identifica la regla del criterio que está usando. A continuación se explica lo que se hace en cada caso:
        1. Valor del agente – Tipo escalar. Esto quiere decir que el cambio que haya ocurrido se aplica en forma numérica sobre el valor anterior de la VMR de la clase *c_unaVMR*.
        2. Valor mío – Tipo escalar. El cambio se aplica en forma numérica sobre el valor anterior almacenado en "valor a comparar".
        3. Valor del agente – Tipo porcentaje. El cambio se aplica en forma porcentual sobre el valor anterior de la VMR de la clase *c_unaVMR*.
        4. Valor del mío – Tipo porcentaje. El cambio se aplica en forma porcentual sobre el valor anterior almacenado en "valor a comparar".
    - Si la diferencia sobrepasa cualquiera de los límites, inferior y superior, ya sea en porcentaje o numérico, el campo "num_var_cliente" de la subestructura *cliente*, que es la que está siendo evaluada, se actualiza con la VMR que se está procesando, de la cual se ejecutará el submodelo.
    - Si el campo de la variable a ejecutar ya está ocupado, es decir, que su valor es diferente de cero, el valor de "cambió más de una variable" se vuelve verdadero. Esto para que la ejecución del modelo sea completa.

- Si la regla es 2 o 4, se actualiza el campo de “valor a comparar” con el valor nuevo que llegó.
- Se actualiza la hora de la evaluación de este cliente.

- Al terminar todos los clientes de una VMR, se actualiza la hora de la última evaluación de esta variable.

5.2.4. Registros de Windows

El proceso descrito antes da como resultado una lista de modelos con su respectiva VMR de la que ha de ejecutarse su submodelo; o en el caso de que se tenga que ejecutar todo, el valor “cambió más de una variable” llega como verdadero. Para comunicar esta información con la ejecución, se creará un registro de Windows donde se almacenará la lista de modelos, su dirección y la variable de la cual se debe ejecutar su submodelo. Los registros de Windows están compuestos de cuatro partes: nombre de la aplicación, sección, clave y ajuste o valor que se almacena. Se crearán dos secciones para la aplicación “FLAG 2014” como se ve a continuación:

Setting("FLAG_2014", "DIRECCION_MODELOS_A_EJECUTAR", consecutivo, valor)

Setting("FLAG_2014", "VARIABLE_A_EJECUTAR", consecutivo, valor)

Se almacenará la misma cantidad de claves como modelos a ejecutar se hayan obtenido. La clave, que cambia conforme la cantidad de modelos, está dada por “*consecutivo”* mientras que la dirección y la variable están en “*valor”*. Los modelos consecutivos inician desde el 1, lo que no quiere decir que se refiere al modelo número 1 sino que es el primero en ejecutarse.

El algoritmo consiste de recorrer el arreglo de instancias del tipo de la clase mostrada en la Tabla 51. Si existe la instancia se procede a leer la clase. Si “cambió más de una variable” es verdadera, el valor que se guarda en la clave “variable a ejecutar” será el número 0, de lo contrario será la “variable a ejecutar”. En la misma iteración se guarda la dirección del modelo que aparece en la clase del modelo. De esta manera, el primer modelo se almacena en la clave 1 de ambas secciones (direcciones y variable), con el valor respectivo en cada una, lo mismo para cada iteración. Cuando termina la lectura de modelos, se guarda otro dato en la clave siguiente:

Setting("FLAG_2014", "VARIABLE_A_EJECUTAR", “cuantos”, valor)

Este es el valor total de modelos que se deben ejecutar en esta sesión de ejecución. Cuando esta clave vale cero, la ejecución no procede hasta que haya modelos que ejecutar.

6. EJECUCION DE MODELOS

6.1. Introducción

La ejecución del modelo de un cliente significa que se recalculan los valores de sus variables (calculadas) con los valores de sus propias variables y los de las VMR que usa su modelo. Este proceso consiste en ejecutar cada una de las fórmulas del modelo del cliente. Como se vio, las fórmulas se ejecutan en orden ascendente de nivel, donde:

Nivel de una fórmula = Max [nivel (operandos)] + 1

Ya se describió el proceso del cálculo de los niveles de las fórmulas, que resulta en lo que se denomina “lista ordenada de fórmulas”.

Es importante señalar que durante el proceso de recálculo de los valores de las variables del cliente, se aplican los criterios de alarma (cuando sea el caso). Este tema se describe en la última sección de este capítulo (Generación de alarmas). FLAG contempla la eficiencia de los procesos puesto que está diseñado para dar servicio a muchos clientes. Para lograr eficiencia en la ejecución de modelos, se tomaron tres acciones:

a) Se elaboró el programa de cálculo de fórmulas de modo que se minimizan las operaciones de I-O necesarias. Por un lado, cuando se usan (leen o actualizan) los valores de una variable, se dejan en memoria de modo que cuando haya necesidad de utilizarlos posteriormente no haya que leerlos nuevamente del archivo. También se ahorran lecturas cuando una variable es aditiva dentro de una fórmula, por lo que solo hay una lectura: los valores de la variable de esa fórmula. (Si hay un operando aditivo, éste ya se ha calculado previamente y ya están sus valores en memoria). De esta forma se aplica el cambio sin tener que acceder a los demás operandos.
b) Se contempla el cálculo de un “submodelo” del modelo, lo que significa que en lugar de recalcular todas las variables del cliente, sólo se recalculan las que se ven afectadas por el cambio de una VMR. Para dicho efecto, se invoca la ejecución basada en esta única VMR.

De este modo, este capítulo contiene secciones por separado para:

- Modo de inicio de la ejecución
- El cálculo de un modelo (completo)
- El cálculo de un submodelo (de una VMR)
- Generación de alarmas

6.2. Modo de Inicio de la Ejecución

El programa que ejecuta los modelos recibe como datos la lista de modelos de clientes que se deben recalcular en una ejecución del programa. Es decir, se invoca el proceso para varios clientes, y el programa ejecuta los modelos correspondientes uno tras otro.

Ya se mencionó que la comunicación entre el motor y el programa de ejecución se hace vía los registros del sistema comúnmente llamados “regwin”. La lista final de modelos se almacena en un registro de Windows. En el capítulo anterior ya se explicó la forma de generar y guardar esta lista: ahora sólo hay que leer los *regwin* y determinar cada modelo que debe ejecutarse junto con su respectiva VMR, si es el caso.

La invocación y ejecución de modelos siempre trabajan de forma simultánea. El programa de ejecución lee los registros de Windows en todo momento, si el valor en la clave “cuantos” es diferente de cero, se procede a la ejecución de modelos. La lectura del registro de Windows se hace de la siguiente forma:

- Se inicia un ciclo desde 1 hasta “cuántos”
- Se obtienen la dirección del modelo y la variable a ejecutar que están en la clave de cada sección en el registro de Windows correspondiente a la actual iteración.
- Si la variable a ejecutar es cero, se llama a la ejecución completa del modelo.
- Si la variable es diferente de cero, se ejecuta el submodelo correspondiente.

De esta forma se ejecuta cada modelo o submodelo, uno a la vez, hasta terminar de leer todas las claves del *regwin*. Los temas que siguen explican la forma de calcular un modelo y un submodelo, además del cálculo de alarmas.

6.3. Estructuras y Arreglos Necesarios

A lo largo de todo el proceso de ejecución se hará uso de ciertas estructuras y arreglos que es preciso detallar ya que de lo contrario el proceso y el cálculo no quedarán del todo claros. Como se ha dicho, hay dos tipos de variables que se pueden encontrar en un modelo FLAG: escalares y aleatorias. La Tabla 54 y Tabla 55 muestran el contenido de las *clases* que se usarán para manejar los valores de cada tipo de variable.

Tabla 54. Estructura de variables numéricas

CAMPO	TIPO	DESCRIPCIÓN
Valores	Valores reales numéricos	Arreglo de los 24 valores que corresponden a cada periodo. Se extraen directamente del archivo.

Tabla 55. Estructura de variables aleatorias

CAMPO	TIPO	DESCRIPCIÓN
Valores	Valores reales distribución	Arreglo de los 24 periodos con sus cuatro valores y probabilidades correspondientes. Se extraen directamente del archivo

Se hace de esta forma porque el manejo de las variables no será consecutivo y habrá la necesidad de usar variables con un número alto sin tener que instanciar las anteriores. La ventaja de usar clases, además de lo anterior, es que al dimensionar un arreglo de las mismas, el espacio en memoria no es el tamaño del arreglo sino solo el de los elementos instanciados. Hacer esto ahorra tiempo en lectura y uso de memoria dinámica.

Las variables de abajo siempre usan alguno de los tipos mostrados en las dos tablas anteriores porque lo único que se necesita de ellas es su valor actual ya que no cambian, a diferencia de las calculadas cuya estructura de la clase se puede ver en la Tabla 56. Se sabe que puede haber variables calculadas de tipo aleatorias, sin embargo usan la clase especificada en la Tabla 55, al igual que si fuese de abajo. La razón es porque a una variable aleatoria no se le puede definir una alarma y los campos de la clase “una variable con fórmula” de la Tabla 56 son usados para esto, así como también para el cálculo del submodelo que se explicará más adelante.

La clase de las variables que tienen fórmula está definida por la estructura mostrada en la Tabla 56. Los campos aquí mostrados se usan en distintas partes del proceso y se irán detallando a medida que avanza este capítulo.

Tabla 56. Información contenida en la clase “una variable con fórmula”

CAMPO	TIPO	DESCRIPCIÓN
Valores nuevos	Flotante	Arreglo de los 24 valores nuevos que corresponden a cada periodo. Son el resultado del cálculo de la fórmula.
Valores anteriores	Flotante	Arreglo de los 24 valores anteriores al cálculo de la fórmula.
Factor de aplicación	Byte	El factor aplicado a los valores de esta variable
Periodo actual	Byte	Periodo que está siendo actualizado. Viene de FLAG.
Cambios en valores	Flotante	Arreglo de 24 valores con la diferencia entre los valores nuevos y los anteriores.
Tiene alarma	Boolean	Verdadero si a esta variable se le definió alarma.

Alarma de esta variable	Variables con alarma	Datos de la alarma que el cliente creó para esta variable.
Gravedad acumulada	Entero	Puntos acumulados por los periodos
Periodo a reportar	Byte	Periodo de gravedad máxima.

Ahora que ya se conocen las clases utilizadas en este proceso, lo siguiente es detallar los arreglos que se usarán para sus instancias. En total se definen cinco arreglos de clases: dos del tipo escalar (una para las VMR y otra para VCL), dos aleatorios y un arreglo para las fórmulas. Las dos primeras son del tipo señalado en la Tabla 54, las siguientes dos como la Tabla 55 y el arreglo de clases tipo fórmula está representado por la Tabla 56. Observación: se tiene que separar las instancias de la misma clase puesto que se accede al elemento del arreglo por número (módulo 1000,000, como siempre) de modo que se pueden repitir índices; una VCL número 11 y otra VMR de número 2000,011.

Al iniciar el proceso de ejecución, además de obtener información de cada modelo, se redimensionan los arreglos de las VMR, es decir, el de variables escalares y otro para las aleatorias. Ambos arreglos deben tener un tamaño máximo de un millón (definido en el diseño). Como son arreglos de clases lo único que resta es instanciar cada VMR al momento que sea usada por primera vez y por el primer modelo que la necesite, hecho esto sus valores permanecen en memoria hasta el término del proceso ya que todos los clientes harán uso de ellas.

6.4. Cálculo de un Modelo

La ejecución de todo el modelo de un cliente consiste en ejecutar cada una de sus fórmulas en el orden ascendente por nivel de ejecución. El programa lee las estructuras iniciales del archivo de fórmulas que son el registro base y la lista de variables ordenadas. Con la primera se obtiene el tamaño de la segunda y además la posición del inicio de las fórmulas. Se ejecutan una tras otra en el orden indicado por la lista ordenada.

6.4.1. Preparación del modelo

La preparación se hace siempre al inicio de la ejecución de cada modelo, esto es porque las variables del cliente usadas son diferentes para cada modelo, a excepción de las VMR que las usan todos. Como se hizo con las VMR, se dimensionan (con dimensión 500) los arreglos de las VCL escalares de abajo y VCL calculadas a un millón y VCL aleatorias. Si es necesario (es decir, aparece una variable que resulta en un índice mayor a 500), se incrementa la dimensión del arreglo involucrado con el parámetro preserve). Cada elemento es del tipo de las clases mencionadas y mostradas en la Tabla 54, Tabla 55 y Tabla 56 por lo que el uso de memoria se hará hasta instanciar el de su variable correspondiente.

En este paso se instancian todas las variables calculadas, es decir, las que están en la lista de variables ordenadas del archivo de fórmulas. Las demás no se instancian sino hasta que son usadas por primera vez y después se mantienen en memoria para su uso posterior. Al momento de instanciar cada variable con fórmula, se obtienen sus valores del archivo y se mantienen en memoria (lo mismo se hace con las variables de abajo). Esto se hace así porque sin duda las fórmulas se utilizarán por lo menos una vez a la hora del cálculo.

Las variables calculadas, como ya se dijo, cuentan con otras características adicionales a las de abajo, como son los valores anteriores y los nuevos, sus criterios de alarma, si es el caso y otros datos para la evaluación de los mismos (ver Tabla 56). Al iniciar la ejecución de un modelo se determina si existen alarmas almacenadas en su archivo de fórmulas, de ser así se extraen y se agregan a la instancia de su correspondiente VCL calculada.

6.4.2. Ejecución de una fórmula

Una fórmula inicia su ejecución con la extracción de la misma del archivo. Se determina el número de byte donde se encuentra a partir de la lista ordenada (es un dato de los elementos de esa lista). De hecho, se pueden leer en la posición siguiente a la anterior, puesto que se graban en el orden en que aparecen en la lista ordenada de fórmulas.

Se ejecuta una rutina con nombre “ejecuta esta fórmula” y se mandan como parámetro, y por referencia, los valores nuevos (ver Tabla 56) que están en la instancia de “una variable con fórmula” de su correspondiente número. Lo anterior quiere decir que se ejecuta la fórmula y al terminar, los valores del cálculo se almacenan en “valores nuevos” que se devuelven por referencia. En seguida se aplican los criterios de alarma ya que ahora se tienen los valores nuevos y los valores anteriores (extraídos inicialmente del archivo) para hacer la evaluación del cambio. Lo concerniente a las alarmas se expone en la última parte de este capítulo.

6.4.3. Cálculo de subtotales

La ejecución de una fórmula consta de la ejecución de cada uno de sus subtotales, siendo el último de ellos el valor nuevo de la fórmula.

Se creó un arreglo de control para almacenar los valores de los subtotales que se vayan calculando, esto porque cuando en una fórmula, hay más de un subtotal, el primero se usa en alguno de los siguientes; lo mismo para el segundo y el tercero. Dicho arreglo es del tamaño del máximo de subtotales de la actual fórmula. Al ir

calculando cada uno, sus valores se almacenan en el arreglo para su uso posterior.

Un subtotal consta de un número máximo de operandos, valor que está almacenado en la estructura del subtotal. Se lee cada uno de ellos y dependiendo del tipo que sea (VCL, VMR, sinónimo o subtotal) se verifica si ya está instanciado. En caso de ser variable calculada su instancia ya existe porque éstas se instanciaron al inicio del proceso. En caso de ser de abajo o una variable aleatoria, se verifica también si ya está instanciada, de ser así se continúa el proceso, de lo contrario se procede a instanciarla.

Sobra decir que en el primer subtotal nunca aparecerá un tipo de operando "subtotal" ya que no hay ninguno antes de éste. Como se mencionó, hay un arreglo de control de subtotales, por lo que cuando se necesita alguno, éste ya está en memoria en dicho arreglo; basta saber a cuál número de subtotal se refiere el operando; este dato está almacenado en el campo donde se guardara el número de variable, en este caso es el número del subtotal a usar.

FLAG tiene una restricción en el cálculo de un subtotal: las operaciones se hacen únicamente de a dos operandos a la vez. Los valores se van acumulando en una variable temporal (un elemento del arreglo de control de subtotales). Cuando inicia el cálculo del subtotal, los valores del primer operando se asignan a la variable de control aplicando la operación correspondiente. Al leer el siguiente operando el resultado del subtotal se acumula en la misma pero aplicando la operación del operando que sigue,

```
For i = 1 to num_opdos
        subtotal (j) = [subtotal (j)] [operación(i)] [operando (i)]
Next i
```

donde:

num_opdos: total de operandos del actual subtotal
subtotal (j): subtotal que se está calculando
operando (i): operando que se está aplicando al cálculo
operación (i): operación correspondiente al operando (i), puede ser suma, resta, producto, división.

De esta forma el resultado del subtotal se va acumulando en una sola variable hasta leer todos los operandos.

En la Tabla 11 se puede ver que hay más operaciones que se pueden usar en las fórmulas además de las cuatro básicas aquí mencionadas. A continuación se mencionan unos puntos importantes que se deben tomar en cuenta para la ejecución que se está explicando:

- Si el operando inicial es de tipo escalar, la operación que se puede aplicar es solamente resta (le cambia el signo a los valores del primer operando),

mínimo, máximo, raíz y cuadrado. Esto es porque el resultado de las mismas es otro escalar. Se decidió que no se aplicarían a distribuciones por el hecho de tener probabilidades, y en general carecerá de sentido usar la raíz cuadrada del valor.

- Las operaciones máximo y mínimo son viables sólo si todos los operandos del subtotal son escalares ya que una distribución tiene cuatro valores con cuatro probabilidades. Si un subtotal tiene una operación tipo mínimo o máximo, no puede tener otras operaciones.
- Por el contrario, las operaciones estadísticas sólo aplican a variables aleatorias por razones obvias. Sin embargo, el resultado es un escalar, lo cual debe tomarse en cuenta en el tipo de subtotal que se está calculando.

6.4.4. Cálculo de un subtotal aleatorio

Para que el resultado de las operaciones dentro de un subtotal sea de tipo aleatorio, por lo menos uno de sus operandos debe ser de este tipo. Además, si es el primer operando, no debe tener aplicada una operación estadística por lo que ya se dijo, el resultado sería una variable escalar.

Un subtotal aleatorio puede tener operandos escalares, sin embargo, ninguno de ellos puede ser el inicial ya que la variable de acumulación es del tipo del subtotal y al iniciar el cálculo, a esta variable se le asigna el valor del primer operando. La ejecución no se preocupa de estos detalles puesto que eso es trabajo de la captura; cuando se agrega una fórmula el programa revisa cada subtotal y si alguno está mal capturado se avisa al cliente para que lo rectifique.

Finalmente, el cálculo del subtotal se obtiene ejecutando la operación entre dos variables aleatorias, o bien, una aleatoria y una escalar. En el subcapítulo *1.3 Probabilidades discretas* se explica detalladamente la forma en que se efectúan tales las operaciones por lo que no se repetirá aquí dicho proceso; basta decir que se usa ese algoritmo en el programa de cálculo y el resultado se acumula en la variable de control del subtotal para su uso posterior.

6.4.5. Cálculo de un subtotal escalar

Un subtotal escalar sólo puede tener un operando de tipo aleatorio si éste es el primero y corresponde a una operación estadística. Esto es porque la variable de acumulación es de tipo escalar y al iniciar, el resultado de esta primer operación, también es escalar. El cálculo se efectúa de forma normal, es decir, entre dos operandos y se va a cumulando el resultado en el subtotal hasta leer todos los operandos.

6.5. Cálculo de un Submodelo

La ejecución de un submodelo o una lista de fórmulas que son afectadas por el cambio de una VMR, se efectúa leyendo la lista correspondiente a dicha variable en el archivo plano, en el apartado de listas de submodelos. El proceso se realiza de la siguiente forma:

- El programa de ejecución lee, del archivo de fórmulas, la lista "índice" de las VMR "con submodelo", almacenada inmediatamente después de las fórmulas y la guarda en memoria;
- La ejecución recibe como parámetro la VMR a procesar, así que la busca en la lista "índice", obteniendo así su posición en el arreglo de listas de fórmulas; como las VMR aparecen en la lista en orden númérico, se puede hacer una búsqueda binaria, aunque no vale la pena: habrá pocas de estas VMR con submodelo, y el proceso se hace una sola vez para el cliente;
- Se carga a memoria el arreglo de fórmulas a ejecutar como parte de ese submodelo; se instancia cada una de ellas en la clase "una variable con fórmula" obteniendo así sus valores anteriores del archivo;
- Al igual que en una ejecución completa, se cargan e instancian las alarmas para su proceso; en este caso solo para las variables que son afectadas;
- Se recorre la lista de fórmulas para su ejecución, que ya está en orden ascendente de nivel. Para cada elemento se hace lo siguiente:
    - Si la hija que la invocó es 0, la ejecución de esa fórmula es de forma normal, es decir, se ejecutan las operaciones con todos los operandos que intervienen.
    - Cuando es diferente de cero, lo primero es almacenar el signo de la hija en una variable temporal. Se hace en una variable de tipo entero y se guarda 1 si es sumando o -1 si es sustraendo.
    - Se calcula el valor nuevo de la fórmula. Para esto se recorren los 24 periodos, en cada uno el valor nuevo es igual al anterior más el cambio en la variable hija multiplicado por el signo, que se obtuvo antes:

$$Form.valorNuevo\ (i) = [Form.valorAnterior\ (i)] + (signo) * [Hija.valorCambio\ (i)]$$

donde:

Form: fórmula que se está calculando
Signo: signo con el que llegó la hija
i: periodo que se está calculando
Hija: variable hija con valores nuevos conocidos y, por consiguiente, también se conocen los cambios

El cambio en la variable hija ya fue calculado en la clase “una variable con fórmula” de dicha variable.

- Se guardan los valores nuevos de la fórmula que se está calculando en la instancia de la clase “una variable con fórmula” que le corresponde.
- Se calculan los “cambios” (valor nuevo menos valor anterior) para esta variable ya que serán usados posteriormente cuando ésta aparezca como hija en otra fórmula.
- Se ejecuta lo de alarmas.
- Finalmente, se actualizan los valores de la fórmula en el archivo plano.

6.6. Generación de Alarmas

Los criterios de alarmas se ejecutan simultáneamente con la ejecución de las fórmulas. La rutina encargada de esto se ejecuta, en ambos tipos de ejecución (de un submodelo o todo el modelo), justo después de calcular el nuevo valor de la fórmula. Se ha mencionado previamente que Velázquez (2009) en su trabajo de tesis y en un artículo de Bauer y Velázquez (2010) describió este módulo. Sin embargo, se le hicieron cambios y se reprogramó en el programa de ejecución de modelos todo lo referente a las alarmas, no sólo porque ya no se leen los criterios de la base de datos (sino del archivo de fórmulas) sino también porque se hicieron más eficientes la aplicación de los criterios y la determinación de la contribución de los cambios a la gravedad, que es lo que determina si procede una alarma o no.

Las variables, de las cuales se evaluará o no su criterio de alarma, son las que se inicializan en la clase, ya mencionada, “una variable con fórmula”. Esta clase se instancia para cada variable escalar calculada que es afectada en una ejecución que, en ocasiones serán sólo las de la lista del submodelo y a veces todas las calculadas. Sin embargo, los criterios de alarma estarán definidos solo en algunas de éstas variables calculadas, las que el cliente señalo como críticas. En general no serán más de cinco, aunque esto no constituye una restricción.

El cálculo de fórmulas utiliza los campos: valores nuevos, valores anteriores y cambios en valores.

Los criterios de alarma están almacenados en el archivo de fórmulas. Ya se mencionó que antes de iniciar la ejecución se extraen los criterios, para lo cual se lee del registro base el número de variables que tienen alarma. Si hay una o más con alarma, se extrae la información general para este cliente.

Tabla 57. Estructura de alarmas de un cliente: “Información general”.

CAMPO	TIPO	DESCRIPCIÓN
E-mail	Texto	Correo electrónico registrado por el cliente. Aquí se le enviará el aviso.
Teléfono	Texto	Teléfono registrado para recibir SMS y llamadas de aviso.
Nombre	Texto	Nombre del cliente. Se usa para generar el aviso.
Cuántas variables tienen alarma	Entero	Cantidad de variables calculadas que tienen asignada una alarma.
Puntaje de cada rango	Arreglo de enteros	Puntos que definen, al final de la ejecución, el tipo de aviso que se debe generar: mail, SMS o llamada.

Al iniciar el proceso para un cliente, ya sea que se trate de ejecución completa o de un submodelo, se cargan a memoria los datos de la estructura de “información general” que se encuentra en el archivo plano (ver Tabla 57). Esta estructura permanece intacta durante toda la ejecución del modelo de un cliente. Al terminar, se resetea y se extraen los datos correspondientes al siguiente cliente. Esta información se usa para que el programa de envío de avisos sepa donde mandarla; es decir, si el aviso resulta un mensaje de texto no urgente, tiene el correo electrónico, si es crítico necesita el número de teléfono para un SMS y si es urgente, con el mismo número se hace la llamada.

Después se extraen los criterios en un arreglo de tipo “variables con alarma” (ver Tabla 58), el tamaño está en el registro base. Este tipo de estructura se puede ver que aparece en la clase “una variable con fórmula” mostrada en la Tabla 56 ya que ahí se almacenará dicha información para esa variable.

Tabla 58. Estructura de “variables con alarma”

CAMPO	TIPO	DESCRIPCIÓN
Número de variable	Entero	
Nombre	Texto	
Descripción	Texto	
Tipo de límites	Byte	0: valor absoluto 1: rango en porcentaje 2: rango escalar
Tipo de periodo	Byte	Define cuántos y cuáles periodos de la variable son afectados por la alarma. Hay 7 modalidades.
Periodo cero	Texto	Variable de control para la lectura de criterios de periodos referentes al actual.
Parámetros por periodo	Parámetros críticos	Arreglo de 24 elementos. En cada elemento se guardan los criterios de los periodos correspondientes o los referentes al actual que cambió.

Tabla 59. Estructura de “parámetros críticos”

CAMPO	TIPO	DESCRIPCIÓN
Intervalos que usa	Byte	Define si se usa el rango crítico o de emergencia.
Límite inferior	Entero	Arreglo de dos elementos. Uno es el valor del LI del rango crítico y el otro el del rango de emergencia.
Límite superior	Entero	Arreglo de dos elementos. Uno es el valor del LS del rango crítico y el otro el del rango de emergencia
Puntos	Byte	Arreglo de dos elementos. El primero tiene los puntos que se acumulan cuando se sobrepasa el rango crítico; en el otro los puntos para el rango de emergencia.

La estructura “variables con alarma” guarda la información de una variable, mientras que “parámetros críticos” (ver Tabla 59) guarda los límites de valores y puntos para cada periodo que fue definido. Con el arreglo “variables con alarma” en memoria se prosigue a instanciar las “variables afectadas” en la ejecución. Se agrega la información de criterios de alarma a cada instancia de cada VCL calculada

6.6.1. Determina periodos inicial y final

Dentro de la clase de una variable afectada aparece la rutina que ejecuta sus alarmas, ésta se invoca después de calcular los nuevos valores de la variable en cuestión. Lo primero que se hace es calcular el periodo inicial y final que se usaron para la alarma. Esto lo define el “tipo de periodo” que, como se mencionó en el capítulo de clientes, tiene 7 modalidades:

1. Uno periodo
2. Todos los periodos
3. Rango
4. Lista
5. Periodo actual
6. Rango respecto al actual
7. Lista respecto al actual

Para determinar el periodo inicial en cada uno de los casos, se debe leer el valor de “periodo cero” en “variables con alarma”. En el caso 1 el valor del periodo inicial es justamente el que se guardó en “periodo cero”; el periodo final es el mismo ya que es un criterio para un solo periodo. En el caso 2 no es necesario usar este campo, el periodo inicial es el 1 y el final el 24. Para el caso 3 se guardan los dos, periodo inicial y final en “periodo cero” dividido por una diagonal, por lo que se extrae el valor *string* completo, después se lee de la diagonal a la izquierda para obtener el periodo inicial y hacia la derecha para el periodo final. Para la lista, el periodo inicial es 1 y final 24.

Para los siguientes tres casos se hace uso de un valor que debe especificarse a la hora de invocar la ejecución, este es el periodo que ha cambiado que se denomina

como el “actual”. Cuando es un periodo actual (caso 5) el que debe evaluarse, se lee el valor del “periodo actual” en la clase de esta variable, entonces el periodo inicial es éste al igual que el final. En caso 6 y 7 se hace lo mismo que en el caso 3, se extrae el valor del *string* en “periodo cero” que está dividido por una diagonal, a la izquierda de la misma está el periodo inicial y a su derecha el periodo final. En ambos casos se trata de un rango de periodos, solo que para el caso 7 cada uno tiene un criterio diferente. Los valores que se obtienen no son el periodo en sí sino su referencia al actual, por esto solo basta con sumar dicho valor al que llegó como actual y se obtendrán los verdaderos periodos inicial y final.

En los tres últimos casos se hace un movimiento de la información de los criterios que están en “variables con alarma”. Como se explicó a la hora de guardar el archivo, el criterio en el caso 5 se guarda en la primera posición del arreglo de “parámetros críticos”, por ser dinámico el periodo actual. Para los casos 6 y 7, los criterios se guardan desde la posición cero hasta el máximo de periodos utilizados en esa variable. Así pues, determinados los valores de “periodo inicial” y “periodo final”, se mueven de posición, en el arreglo “parámetros críticos”, al que les corresponde. Por ejemplo, si hay 5 criterios guardados desde la primera posición hasta la 5 y el periodo inicial guardado es -1, periodo final 3 y el periodo actual es 19, entonces los cuatro criterios se mueven de sus posiciones iniciales y se graban a partir del periodo 18 al 22 (ver Figura 107).

Posición inicial de los criterios en el archivo

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Criterios reubicados

																	-1	0	1	2	3		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	**19**	20	21	22	23	24

Figura 107. Determinación de rangos de periodos respecto a la variable actual 19

6.6.2. Cálculo de gravedad acumulada en periodos

Para calcular y acumular las gravedades de todos los periodos involucrados en la alarma se inicia un ciclo desde el periodo inicial hasta el final. En cada uno se leen los valores de sus criterios, es decir, los límites inferior y superior para los rangos crítico y de emergencia, cuando sea el caso. A continuación se lee el tipo de límite definido en esa alarma. En la sección de criterios de alarmas del capítulo de clientes se explicó cada uno de los tres tipos:

1. Absoluto
2. Porcentaje
3. Amplitud

Para el segundo caso se convierten los porcentajes guardados en los límites a los valores que se van a comparar con los nuevos. La conversión se hace de forma sencilla por medio de una regla de tres: se multiplica el valor del límite (en porcentaje) por el valor anterior del periodo y se divide entre 100. Después este valor es restado y sumado a los límites superior e inferior. Estos nuevos valores de los límites son los que se van a comparar.

En el tercer caso los límites superior e inferior se suman y restan, respectivamente, al valor anterior del periodo y estos serán los nuevos valores a comparar. Para el caso 1 no es necesario ninguna operación ya que los valores guardados son los que se van a comparar directamente.

Al tener los límites definidos, se compara el valor nuevo del periodo, primero con los límites de emergencia, para verificar si los supera, es decir, si el valor nuevo es menor que el límite inferior o mayor que el límite superior. Si es el caso, se guardan los puntos definidos para ese rango. De no ser así se compara ahora con los límites críticos y de superarlos se guardarán sus respectivos puntos en vez de los de emergencia. Si el cambio es tan poco significativo que no afecta al criterio, no se obtienen puntos y este periodo no aportará puntos a la alarma.

En la instancia de la clase de la variable con fórmula hay un campo que almacena la gravedad que acumulan los periodos evaluados. Al obtener los puntos que aporta cada uno, se acumulan en la variable “gravedad” y al final se tendrá un valor de puntos totales aportados por los periodos de esa variable. Terminada la evaluación de la variable, los puntos acumulados se comparan con el máximo guardado. Si lo supera, se define a éste como el nuevo máximo y la variable a reportar es la actual. Los puntos de esta variable se acumulan en otra “gravedad acumulada” que se actualiza cada vez que se evalúa una variable afectada. Con los puntos que se acumulan de todas las variables se determinará el tipo de alarma que ha de enviarse.

6.6.3. Generar mensaje de alarma

Al término de una ejecución las variables afectadas ya tienen un puntaje acumulado, la mayor es la que se ha de reportar en el aviso; también está calculado el puntaje total de ese modelo (a partir de la suma de las gravedades de las variables afectadas) para definir el tipo de aviso que ha de ser enviado.

En la información general de alarmas de un cliente mostrada en la Tabla 57 aparece un arreglo de los puntajes que definen el tipo de aviso que generará la alarma. Para cada rango que se sobrepasa el sistema debe tomar acciones:

1. *Cambio en el modelo*. Cuando los puntos acumulados sobrepasan este puntaje, se envía un mensaje de texto por medio de e-mail con la

recomendación de que debe checar su modelo ya que ha habido un cambio.

2. *Cambio significativo*. Cuando hay más puntos acumulados quiere decir que el modelo se modificó significativamente. El aviso es por medio de un mensaje de texto SMS con la recomendación de revisar su modelo.
3. *Cambio crítico*. El modelo ha cambiado mucho y es necesario que el cliente esté al tanto de lo que ha pasado. Se envía un SMS con la variable crítica, periodo y valores que han cambiado.
4. *Situación de emergencia*. La acumulación de puntos es bastante elevada por lo que el modelo tiene cambios que es urgente que el cliente sepa a la brevedad. El aviso es por medio de una llamada telefónica por parte del personal de FLAG que se encarga del monitoreo de los modelos de los clientes. El aviso consta de informar al cliente la situación de su modelo, así como la o las variables que han sido afectadas y tuvieran cambios importantes en la última ejecución.

Antes de generar el aviso el programa debe obtener la información necesaria para la alarma, la cual incluye lo siguiente:

- Variable crítica
- Nombre del cliente
- Periodo a reportar
- Valor que se reporta
- Gravedad acumulada (puntaje máximo)
- Periodo que se está actualizando

Se compara el puntaje que acumuló la ejecución del modelo con los rangos establecidos inicialmente por el cliente y que, como se mencionó, están en “información general” de la alarma. El puntaje se compara con el rango máximo, es decir, el caso 4 de situación de emergencia; si lo sobrepasa, el tipo de aviso es de tipo llamada. Si no lo sobrepasa se continúa con el siguiente y así sucesivamente hasta el de menor puntaje. Si el cambio es tan mínimo que no sobrepase el primer rango, no se generará ningún tipo de aviso y se continuará con la ejecución del modelo del siguiente cliente.

A continuación se describen los mensajes “tipo” para cada uno de los tres primeros casos ya que el último no es un texto sino una llamada telefónica.

1. “Su modelo ha cambiado, favor de hacer una consulta para verificar cambios en variables.”
2. “Estimado señor [NOMBRE] su modelo ha cambiado de forma significativa, favor de hacer una consulta para verificar cambios en variables.”

3. “Estimado señor [NOMBRE] su modelo de FLAG ha cambiado de forma crítica. La variable [VARIABLE CRÍTICA] ahora tiene el valor [VALOR NUEVO DE LA VARIABLE] y lo tomó en el periodo [PERIODO CRÍTICO].”

Como se puede ver, los mensajes son sencillos. En el primer caso solo se exhorta al cliente a checar su modelo sin más preocupación. En el segundo caso ya es algo más importante por lo que, al menos, causará curiosidad en el cliente para verificar los cambios que han ocurrido. El tercer caso ya es un aviso de cambio crítico, se le menciona al cliente la variable más crítica así como el valor que ha tomado y el periodo más crítico modificado, que la mayoría de las veces es el actual.

Tabla 60. Tabla de “alarmas generadas”

CAMPO	TIPO	DESCRIPCIÓN
ID cliente	Entero	
Nombre	Texto	
E – mail	Texto	
Teléfono	Texto	
Dirección	Texto	
Fecha y hora de la ejecución	Date	Fecha y hora de la ejecución que generó esta alarma
Modo	Byte	Modo de aviso de la alarma
Mensaje	Texto	Mensaje generado
Mensaje enviado	Boolean	Falso hasta que se envía
Fecha y hora del envío del mensaje	Texto/date	
Hubo error	Boolean	Si hay algún error en el envío del SMS o el mail, aquí se reporta para el reenvío
Tuvo respuesta	Boolean	Cuando se haya recibido el aviso
Última llamada intentada	Texto/date	Se reporta la fecha y hora del último intento de comunicación con el cliente por teléfono

Cada vez que se genera una alarma de un cliente, la tabla de “alarmas generadas” adquiere un nuevo registro con la información mostrada en la Tabla 60. Se puede ver que aparece la información principal como nombre, número telefónico y correo electrónico, con lo que se sabe a quién se envía el aviso. Los demás campos son para control del programa de ***envío de alarmas*** que se describirá en su capítulo más adelante. Dichos campos tienen un valor default, por ejemplo “mensaje enviado”, “tuvo error” y “hubo respuesta” se inician como falsos; la hora del envío del mensaje así como la de la última llamada efectuada permanecen vacías hasta que se realiza el envío del aviso o la llamada. El modo, el mensaje generado y la fecha de la ejecución son datos que se conocen y se mantienen constantes.

7. COMPROBACIÓN DE LAS MEJORAS IMPLEMENTADAS

Simulación para cuantificar la reducción de las duraciones de los procesos

Se formularon dos modelos para determinar el impacto de las mejoras sobre la duración de la ejecución de modelos: Mod1 con 44 variables y 33 fórmulas y Mod2 con 332 variables y 264 fórmulas. En ambos se incluyeron VMRs con sinónimos.

Se ha mencionado que uno de los aspectos que se contemplaron para reducir el tiempo de proceso era minimizar las operaciones de entrada y salida. Cuando se usan los valores de una variable (como variable a calcular o como operando) se dejan en memoria para evitar tener que obtenerlas nuevamente de disco cuando aparezcan como operandos en otra fórmula en forma de sinónimos. Esto es lo que se indica como “guardar valores ya leídos”.

Puesto que las ejecuciones son demasiado breves para comparar los algoritmos (se obtienen en milisegundos) se invocaron ambos modelos 1000 veces. Además se determinó la duración de ejecutar el Mod1 500 veces y el Mod2 500 veces (para reflejar una situación en la que hay que recalcular modelos de distintas dimensiones). Para comprobar que la duración de N ejecuciones es lineal, se ejecutaron los modelos 500 veces y las duraciones eran prácticamente la mitad.

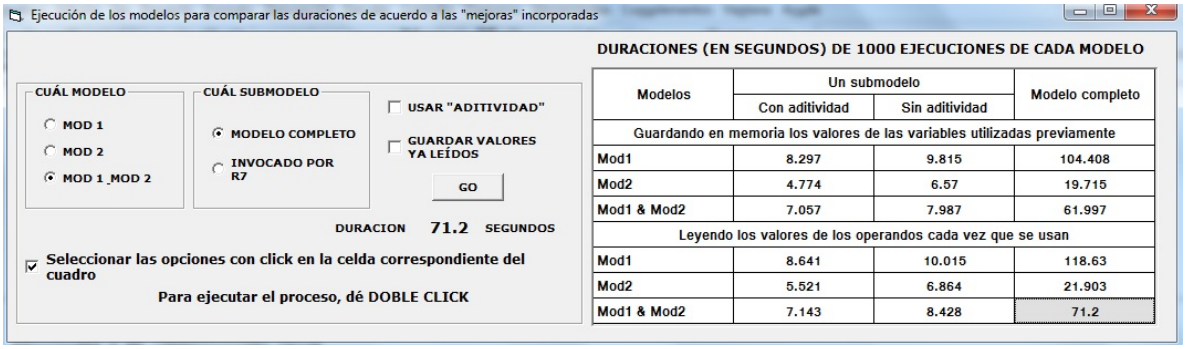
La Figura 108 ilustra la forma con la que se invocaron las ejecuciones de los modelos usando los diversos algoritmos. Se parte de uno que se podría llamar anterior (no se guardan los valores de las variables cuando se usan y se ejecuta el modelo completo). Se le agregan estas dos “mejoras” y cuando se trata de ejecutar submodelos, se comparan las duraciones obtenidas incluyendo el uso de los operandos aditivos con las resultantes de calcular siempre toda la fórmula.

Las duraciones (en segundos con 3 decimales) que se muestran en la Figura 108 se midieron desde el momento en que inicia el cálculo del primer modelo hasta el fin del proceso de todos los modelos. Para determinar el impacto de cada una de las mejoras incorporadas a la nueva versión, se repitieron los procesos incluyendo y excluyendo cada una de ellas.

El operador del programa selecciona el modelo y las mejoras que desea e invoca la ejecución del modelo 1000 veces. Cuando finaliza este proceso, el sistema muestra la duración del último proceso en el lugar adecuado del cuadro.

Este programa se puede usar de dos modos. Se indican las opciones en los campos (la mitad izquierda de la forma) o se da un click sobre la celda del cuadro que se desea: esto resulta en que aparezcan las opciones seleccionadas en los

campos de la izquierda. Un doble click en una de las celdas invoca la ejecución de los modelos.



Modelos	Un submodelo		Modelo completo
	Con aditividad	Sin aditividad	
Guardando en memoria los valores de las variables utilizadas previamente			
Mod1	8.297	9.815	104.408
Mod2	4.774	6.57	19.715
Mod1 & Mod2	7.057	7.987	61.997
Leyendo los valores de los operandos cada vez que se usan			
Mod1	8.641	10.015	118.63
Mod2	5.521	6.864	21.903
Mod1 & Mod2	7.143	8.428	71.2

Figura 108. Forma utilizada para invocar las ejecuciones de las fórmulas de los modelos con las mejoras incorporadas para reducir las duraciones de los procesos. Para cada ejecución se muestra su duración y se agrega al cuadro comparativo.

No hay motivo para pensar que los modelos construidos son representativos, pero sí que los cambios en duraciones serán similares para otros modelos. De ese modo, las reducciones que se mencionan a continuación son aproximadas, además de que no están debidamente comprobadas. Pero indican claramente que hay disminuciones atribuibles a cada uno de las 3 mejoras:

Guardar en memoria los valores de las variables que se utilizan para no tener que leerlos nuevamente: esto reduce en aproximadamente 10% la duración.

Procesar un submodelo en lugar del modelo completo:

- Si se usa la aditividad, la reducción es del orden de 80% (aunque en modelos mayores es un poco mayor);
- Si no se usa la aditividad, la reducción es un poco menor: tarda aproximadamente 10% más que con esta funcionalidad.

CONCLUSIONES

Se anunció que la nueva versión tendría varios aspectos: se completarían funciones que no estaban totalmente implementadas; se mejorarían y simplificarían las interfaces; se eliminaría la necesidad de que el cliente incluyera sinónimos en sus modelos, dejando esto al sistema mismo; se harían más eficientes todos los procesos, especialmente el de ejecución de los modelos; y se agregarían algunas funcionalidades que surgieron como reacción al uso de la versión anterior.

Las simulaciones indican que se logró el objetivo principal, que era reducir significativamente las duraciones de las ejecuciones. Por un lado, los cambios hechos para evitar la lectura repetida de valores de operandos por sí solos redujeron las duraciones en aproximadamente 10%. El uso de los submodelos naturalmente produce una reducción mucho mayor, en especial cuando el modelo tiene operandos aditivos o cuando el modelo contiene muchas fórmulas. De ese modo, esta versión del paquete FLAG es mucho más eficiente que su predecesora, lo que a su vez significa que se podría ofrecer el servicio a un número mayor de clientes.

Un examen de la nueva versión arroja otros resultados favorables: las interfaces son mucho más claras y prácticas que en la versión que se reemplaza; se terminaron algunos módulos, y se implementaron otros que estaban anunciados pero nunca se habían programado.

Se implementó completamente el motor del FLAG, desde la obtención de valores actualizados por agentes hasta el envió de alarmas cuando procedan. Adicionalmente estos procesos en su conjunto se diseñaron y programaron de tal modo que aún con un número elevado de clientes, el servicio funcionaría (no habría cuellos de botella o procesos que no se pudieran realizar por no haber concluido uno anterior).

Los clientes ahora disponen de más consultas, pero especialmente ya no tendrán que lidiar con los sinónimos, puesto que éstos son transparentes para los clientes.

Los algoritmos utilizados para crear y ejecutar los submodelos son aplicables a otro tipo de modelos, lo que le da generalidad al trabajo realizado. La naturaleza de los modelos que permiten utilizar submodelos reside en su estructura, y no depende del tipo de fórmulas o de las operaciones que se implementan.

De ese modo, se puede afirmar que el sistema FLAG está “casi listo” para que alguien ofrezca un servicio basado en él. Hay algunos componentes adicionales que se agregarán, que se mencionan como parte de investigación futura.

Entre los temas están: ampliar la naturaleza de las fórmulas para abarcar modelos que no se pueden formular en el FLAG, lo que ampliaría su aplicabilidad. También se contempla incluir consultas (y quizá ciertas actualizaciones) con programas que funcionen en páginas de internet. Esto tendría un impacto menor al esperado, puesto que los clientes tienen los programas para usar sus modelos, y éstos pueden funcionar con datos que residen en la web. En cambio poder consultar su modelo desde un dispositivo móvil podría tener un impacto mayor.

En especial le falta a la versión actual cierto grado de parametrización, pero esta carencia estuvo planificada desde el inicio. Cuando estuvieran listos los programas se agregarían dispositivos que permitieran limitar los conceptos y funciones en alguna instancia de uso del FLAG. Por ejemplo, si no se necesitan variables aleatorias, podría resultar molesto que le preguntaran a los usuarios si la variable que están creando es aleatoria o no. O si no se ofrecen alarmas, que no les pregunten a sus clientes si desean incluir los criterios. En otras palabras, que resultara posible bloquear algunas opciones y funciones del sistema si las circunstancias indicaran que no usarán.

No se ha desarrollado el módulo denominado “contabilidad”, que incluye la determinación de los costos de los diversos servicios que ofrece FLAG a sus clientes. En particular se deben establecer costos de la consecución de valores por medio de agentes, las ejecuciones de los modelos, el costo de envío de las alarmas y el servicio en general, es decir incluir a un cliente y proveer los servicios a cada uno de ellos.

Finalmente, los paquetes de Bauer en general deben ser traducibles. Esto significa que se pueden traducir todos los elementos que intervienen en alguna interfaz a cualquier idioma sin necesidad de alterar programas o bases de datos. Incorporar esta característica incrementa notablemente la carga de implementación de un paquete, de modo que se decidió postergar la traducibilidad: se introducirá posteriormente si se determinara que podría – una vez más – incrementar la aceptación y por ende el uso del software elaborado.

BIBLIOGRAFÍA

Array formulae. (2008). Recuperado Febrero 16, 2014 de http://www.decision models.com/optspeedj.htm

Arthur W. Burks, Don W. Warren and Jesse B. Wright. (1954). *An Analysis of a Logical Machine Using Parenthesis-Free Notation.* Mathematical Tables and Other Aids to Computation, Volume 8(46): 53 – 57, abril, 1954

Bauer Mengelberg, J. R. (2008a).Efecto de cambios en datos externos a la empresa sobre modelos de flujo de efectivo. Ponencia en el XVIII Coloquio Mexicano de Economía Matemática y Econometría, Facultad de Economía, UNAM.

Bauer Mengelberg, J. R. (2008b). Mecanismos para la ejecución parcial de los modelos afectados por cambios en un dato externos (en un servicio proporcionado a muchos clientes). Ponencia en el XVIII Coloquio Mexicano de Economía Matemática y Econometría, Facultad de Economía, UNAM.

Bauer Mengelberg, J. R. & Velázquez, G.D. (2010). *Timely informing clients of the impact of changes in their business environment.* Issues in Informing Science and Information Technology, Volume 7: 377 – 392, 2010

Berrocal, J. L., et al (2003). Revista española de Documentación Científica, Vol 26, No 1. Recuperado junio 10, 2014 de http://redc.revistas.csic.es/index.php/redc/article/viewArticle/130

Campbell, S.., et al. (2003).101 Microsoft Visual Basic .NET Applications, Microsoft Press.

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, Recuperado de http://elicohen.info/uglyduckling.pdf

Deco, C., et al (2013). Teoría de Bases de Datos. UNR. Rosario, Argentina Recuperado junio 10, 2014 de http://www.dsi.fceia.unr.edu.ar/downloads/base_de_datos/Introduccion.pdf

Díaz, L. S. A. (2009). Obtención de valores del entorno empresarial de clientes de un servicio de información. Tesis (Maestro en ciencias). Montecillo, Texcoco, Edo. de México, Colegio de Postgraduados.

Excel's Smart Recalculation. (2008). Retrieved June 4, 2014 from http://www.decisionmodels.com/calcsecrets.htm

Gackowski, Z. J. (2005). Informing Systems in business environments: a purpose - focused view, Informing Science Journal, Informing Science Institute, Sta. Rosa, California, Volumen 8, 109-111.

Gill, T.G. & Bhattacherjee, A. (2007). The Informing Sciences at a Crossroads: The Role of the Client. Informing Science Journal, Vol. 10. 17-39. Retrieved March 21, 2014 from http://inform.nu/Articles/Vol10/ISJv10p017-039Gill317.pdf

Hípola, P., Vargas – Quesada, B. (1999). "Agentes inteligentes: definición y tipología. Los agentes de información". El profesional de la información.

HYNES, Richard. (2003). Programación de bases de datos con Visual Basic.NET, 1ª Edición, Pearson Educación

Izar, L. J. M. (1998). Elementos de métodos numéricos para Ingeniería. UASLP. San Luis Potosí, México.

Katcheroff, P. (2008). Desarrollador .NET (adapt), Manuales USERS, 1ª Edición, Gradi.

Klush, M. (1999). Intelligent Information Agents. Springer – Verlag

Laha, R. G. & Rohatgi, V. K. (1979). Probability theory. Editorial Wiley, New York, EU. pp 150, 151

Lara N., P y Martínez U., J. A. (2004). Agentes inteligentes en la búsqueda y recuperación de información. Ed. Planeta UOC. Catalunya, España. Recuperado junio 10, 2014 de http://eprints.rclis.org/7941/1/2004-Lib-Agentes.pdf

Laudon, K. C. y Guercio T., C. (2001).Information Technology and Society, 2ª Edición, Wadsworth Publishing Company

Leu, D., Kinser, C., Coiro, J., Cammack, D. (2004). Toward a Theory of New Literacies Emerging From the Internet and Other Information and Communication Technologies. En Theoretical Models and Processes of Reading, 5ª Edición. International Reading Association

Lipschutz, S. (2004). Estructura de Datos. MCGRAW-HILL. México, DF. Recuperado junio 10, 2014 de http://hdl.handle.net/123456789/537

Microsoft (2014). *Thread Class.* Microsoft Developer Network. Retrieved June 4, 2014 from http://msdn.microsoft.com/en-us/library/system.threading.thread.aspx?-cs-save-lang=1&cs-lang=vb#code-snippet-1

Null, L. and Lobur, J. (2006). The essentials of computer organization and architecture. n.p.: Jones and Bartlett, pp. 741-742.

RAE, (2014). Definición de “sinónimo”. Diccionario de la Real Academia Española (DRAE) en línea. Recuperado Febrero 19, 2014 de http://lema.rae.es/drae/?val= sinónimo.

Rob, P. and Coronel, C. (2009). Database Systems: Design, Implementation, and Management. Boston, MA: Course Technology, pp. 90-91.

Senn, J. A. (1990). Sistemas de información para la administración, 4ª Edición, Grupo editorial Iberoamericana

Said, B., Jentsch, F. & Brunswig, F. (2014). Development of business applications. Sanct Leon–ROT, DE, USA. Retrieved June 3, 2014 from http://www.freshpatents. com/-dt20140515ptan20140137075.php

Varela L., V. (2014). Evaluación de riesgos de plagas y enfermedades en cultivos usando datos climáticos en tiempo real. Tesis (Maestro en ciencias). Montecillo, Texcoco, Edo. de México, Colegio de Postgraduados.

Velázquez C. G. D. (2009). Alarmas en un servicio que proporciona información oportuna e interpretable a sus clientes. Tesis (Maestro en ciencias). Montecillo, Texcoco, Edo. de México, Colegio de Postgraduados.

Weiss, M. A. (1993). Data structures and Algorithm Analysis. Reedwodd City, CA: The Benjamin Cummings, pp. 87 – 91, 133 – 138.

Whitten, J. L., Bentley, Lonnie D., Barlow, Victor M. (1996). Análisis y diseño de sistemas de información, 3ª Edición, España.

ANEXOS

1. Módulo de envío de Alarmas

FLAG se divide en cuatro módulos que funcionan de manera independiente respecto a los demás. Por orden en que se ejecutan en el servicio son:

- Módulo de agentes
- Módulo de evaluación de criterios de ejecución
- Módulo de ejecución de modelos
- Módulo de envío de alarmas.

Hasta este momento se han explicado los tres primeros, con sus cambios y mejoras respecto a la versión anterior. El módulo que resta es el de envío de alarmas a los clientes. Éste fue inicialmente desarrollado por Velázquez (2009), sin embargo, la parte de la captura y evaluación de criterios de alarmas fueron mejorados en el presente trabajo, como se explicó en las secciones 3.6 y 6.6, respectivamente. El proceso que sigue el envío de alarmas se explica a continuación.

Este módulo inicia desde donde queda la ejecución de modelos, lo cual es la actualización y/o inserción de nuevos registros en la tabla de alarmas generadas. Se reprodujo la Tabla 60 en la Tabla 61 mostrada a continuación para mejor comprensión de los envíos.

Tabla 61. Tabla de “alarmas generadas” (copia)

CAMPO	TIPO	DESCRIPCIÓN
ID cliente	Entero	
Nombre	Texto	
E – mail	Texto	
Teléfono	Texto	
Dirección	Texto	
Fecha y hora de la ejecución	Date	Fecha y hora de la ejecución que generó esta alarma
Modo	Byte	Modo de aviso de la alarma
Mensaje	Texto	Mensaje generado
Mensaje enviado	Boolean	Falso hasta que se envía
Fecha y hora del envío del mensaje	Texto/date	
Hubo error	Boolean	Si hay algún error en el envío del SMS o el mail, aquí se reporta para el reenvío
Tuvo respuesta	Boolean	Cuando se haya recibido el aviso
Última llamada intentada	Texto/date	Se reporta la fecha y hora del último intento de comunicación con el cliente por teléfono

El intervalo de tiempo entre ejecuciones de envíos será definido por el administrador del sistema, sin embargo, se debe tomar en cuenta que, cuando el servicio esté funcionando y haya muchos usuarios registrados, la actualización de las alarmas generadas será cada minuto ya que es el tiempo mínimo que tardan los agentes en conseguir valores.

Los datos de la tabla de alarmas generadas que no cambian, después de creado el registro, son: id-cliente, nombre, mail, teléfono, dirección, fecha en que se generó el aviso, modo y mensaje generado. La fecha de ejecución y los datos del cliente sirven para bitácora, es decir, se quedan en la base con el fin de tener registrado quién y cuándo generó un aviso. Estos datos también se usan de otra forma, como se verá a continuación.

Dependiendo del modo de envío es como se hará uso de los campos de la tabla. Si el modo es vía correo electrónico, se usa la dirección de correo y se envía el “mensaje” guardado en el mismo registro. De esta forma el cliente puede consultar su mail y así enterarse del comportamiento de su modelo.

Si el modo es de tipo SMS, se usa el número telefónico y se envía el “mensaje” del registro. Para el caso de la llamada telefónica también se usa este número, solo que el envío es por medio de un mensaje de voz (se hace de forma automática) o manualmente, por un empleado de FLAG.

Cuando el programa de ejecución de los envíos de alarmas se dispone a leer la tabla de alarmas generadas, al obtener un registro, el primer campo en evaluar es “mensaje enviado”; si su valor es verdadero, se descarta y se prosigue con el siguiente registro, ya que el aviso fue ejecutado satisfactoriamente. En cada intento de envío siempre se actualiza el campo “fecha y hora de envío del mensaje”. Después de esto hay dos posibles resultados: que el mensaje se haya recibido o haya fallado. Para esto se usan los campos de la tabla: “hubo error” y “tuvo respuesta”. Si el modo de aviso es un mail o un SMS, se determina si hubo error al entregar el mensaje; esto puede ocurrir al fallar la conexión de internet, caída del servidor del servicio de correo, falta de señal del teléfono, entre otras cosas. En caso de que el modo sea llamada telefónica, se usa el campo “tuvo respuesta”. Si el cliente no atiende la llamada, no hay señal, no tiene saldo en el teléfono, se desvía la llamada, etc., “tuvo respuesta” queda como falso.

Como se dijo, los envíos se harán en un intervalo de tiempo y cada vez que se hagan, los campos mencionados se actualizarán. Si fallan los mensajes, entonces “habrá error” y si falla la llamada entonces “no tendrá respuesta” y, para esto último, se registra la fecha y hora de la “última llamada intentada”. El campo “mensaje enviado” se vuelve verdadero hasta que no haya fallos en el envío.

Nota sobre llamadas telefónicas. Con respecto a este tipo de aviso, se mencionó que puede generarse de dos formas: mensaje automático y manual. Cuando el mensaje se efectúa de forma automática, el aviso se vuelve limitado, es decir, cuando el cliente contesta su teléfono solo recibe el mensaje de voz conteniendo el texto guardado en “mensaje”. En cambio, si se hace manualmente hay la posibilidad de que el empleado de FLAG le dé más detalles de su modelo al cliente, los cuales puede ser las VMR que cambiaron y que afectaron su modelo, sus otras VCL que cambiaron, nuevos valores adquiridos, etc.

El aviso de tipo llamada telefónica aún está en modo de afinación, por lo que posteriormente habrá mejoras para bien del servicio y del cliente potencial que hará uso de él.

El módulo de envíos fue desarrollado por Velázquez (2009) en su proyecto de investigación con más detalle, además de que fue publicado por Bauer y Velázquez (2010) en la revista Informing Science, en caso de que el lector quiera profundizar en el tema.

2. Descripción general de la programación

El sistema se desarrolló en la plataforma Windows, en especial con el lenguaje VB,net para aprovechar su naturaleza de orientación a objetos y las facilidades que proporciona al programador en la elaboración y depuración de los programas.

En particular, se dividieron los elementos en una serie de librerías (DLL) para agrupar clases, rutinas y funciones de modo de independizarlas de las restantes y de ese modo poder reutilizarlas en diversos programas.

Se proporciona una lista de estas librerías con una breve descripción de las clases y sus métodos que implementan. Las librerías mismas están en una carpeta del disco adjunto a esta tesis.

2.1. DLLs del sistema FLAG del Cliente

Librería Bases de Datos. Maneja las conexiones y desconexiones de las bases; incluye procedimientos almacenados para cada BD. Inicia y maneja objetos como datasets, tablas y bindings source.

Librería Herramientas. En esta dll se incluyen las clases y estructuras que se usan en la mayor parte del sistema. Las estructuras son:

- Valores de variables numéricas y aleatorias. Son de dos tipos, una para la extracción directa de valores del archivo y otra con los valores reales, es decir, aplicado el factor.

- Estructuras para el archivo de fórmulas: registro base, lista ordenada, una fórmula, tres estructuras usadas por los submodelos y tres por los criterios de alarmas.

Las clases incluidas son:

- fileFórmulas. Se incluyen todas las rutinas que tienen qué ver con las fórmulas: lectura y escritura de ellas así como lo de submodelos y criterios de clientes.
- fileValores. Lo mismo pero para lo referente a los valores.

Librería Consultas. Incluye estructuras especiales que serán usadas en las rutinas y funciones propias de las consultas. También están las interfaces. Esta librería hace uso de la de bases de datos ya que debe conectarse al archivo "queryCatalogue". También hace uso de la librería herramientas para el uso de estructuras y lectura de valores.

Librería Alarmas. Incluye ABC de alarmas del cliente. Están las interfaces necesarias para ello. Hace uso de la librería de bases de datos.

Librería Dibuja Modelo. Están las rutinas que dibujan el modelo del cliente, así como la interfaz que la muestra.

2.2. DLLs del sistema FLAG del Administrador

Librería Bases de Datos. Maneja las conexiones y desconexiones de las bases; incluye procedimientos almacenados para la BD en FLAG. Inicia y maneja objetos como datasets, tablas y bindings source.

Librería Herramientas – Administrador. Se incluyen clases y estructuras como en el sistema del cliente, pero ahora para el manejo del modelo de las variables del mundo real. Además de lo anterior, se incluyen dos clases para su uso en la evaluación de criterios: unaVMR y unCliente.

Librería Tabla de criterios de clientes. Aquí se incluye la lectura de las bases de datos de todos los clientes incluidos en FLAG, con el fin de llenar la tabla de los criterios de evaluación, la cual después será convertida en un "archivo de criterios" para cuando se esté ejecutando el servicio completo de FLAG.

Librería Diagnóstico de FLAG. Se encarga de actualizar la tabla de clientes en la base de FLAG. Para esto debe revisar los archivos y bases de datos de los clientes actuales y, si se han agregado nuevos, agregarlos a la tabla (que será leída en la evaluación de criterios).

Librería Envío de Alarmas. Se encarga de leer la tabla de "alarmas generadas" y enviar los avisos generados para cada cliente especificado.

Librería Herramientas – Cliente. Contiene las estructuras y clases para el manejo de los valores y el archivo de fórmulas de los clientes. También están las clases usadas en la ejecución de modelos, es decir, las de variables escalares, aleatorias y afectadas.

Librería Ejecución de Modelos. Es la dll más robusta en todo el sistema. Contiene las rutinas encargadas de la ejecución de modelos, lo cual incluye el cálculo de fórmulas y subtotales y la generación de los avisos. Utiliza las herramientas del cliente.

3. Bases de datos típicas en el servicio FLAG

A continuación se detallan las bases de datos usadas por FLAG. Son cuatro, dos para el FLAG del cliente y dos para el FLAG administrador.

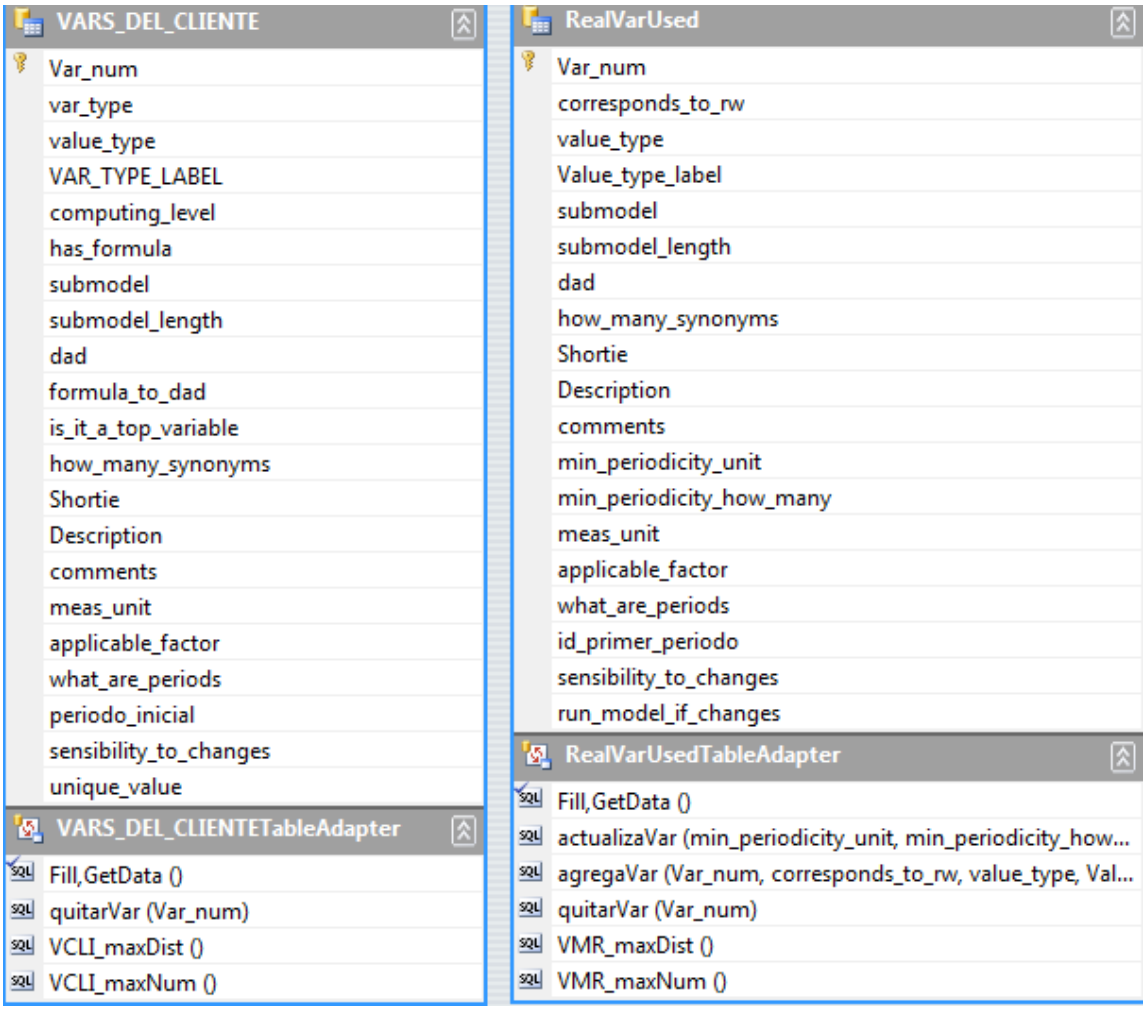


Figura 109. Bases típicas: modelo del cliente. Tablas: variables del cliente y variables del mundo real usadas por el cliente.

Estas bases se almacenan en las carpeta correspondiente al cliente que la está usando, es decir, en su máquina dentro de la carpeta de instalación del FLAG habrá otra llamada “FLAG - Bases de Datos”, aquí se almacena la base que contiene la información del modelo, así como la base de las consultas catalogadas.

Las bases que usa el cliente son:

- Modelo del cliente. El archivo se denomina “modeloBase.accbd”
- Consultas catalogadas. El archivo se denomina “querycatalogue.accbd”

En la Figura 109 se puede ver la tabla de variables del cliente y la de las variables del mundo real que usa este cliente. Se podría decir que son las que contienen la información esencial del modelo, aunque las demás son también importantes.

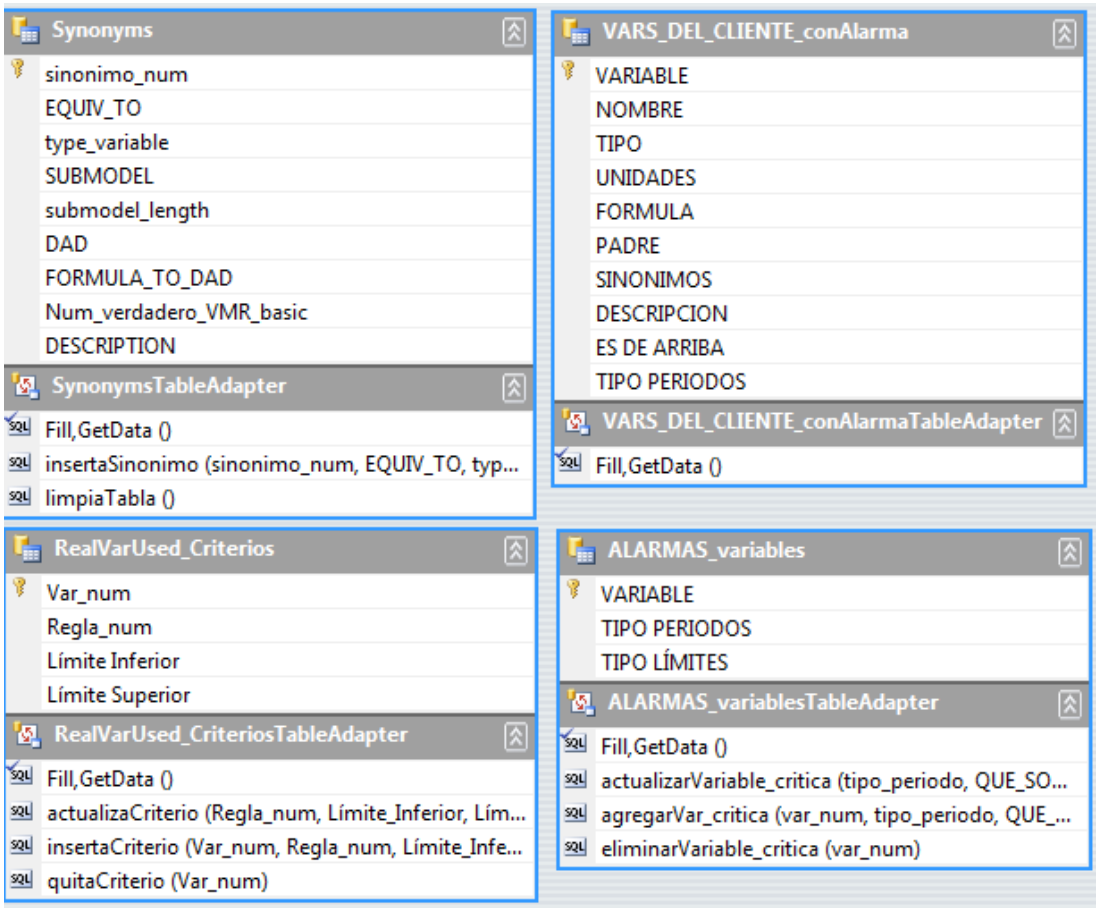


Figura 110. Bases típicas: modelo del cliente. Tablas: sinónimos, variables con alarma, VMRs con criterio y alarmas de variables.

Las tablas que almacenan los criterios de ejecución, alarmas y sinónimos, se muestran en la Figura 110. Y en la Figura 111 se puede ver la tabla de alarmas por periodo y la tabla del cliente, donde se almacena la información principal de cada cliente, así como los puntajes para los avisos de las alarmas.

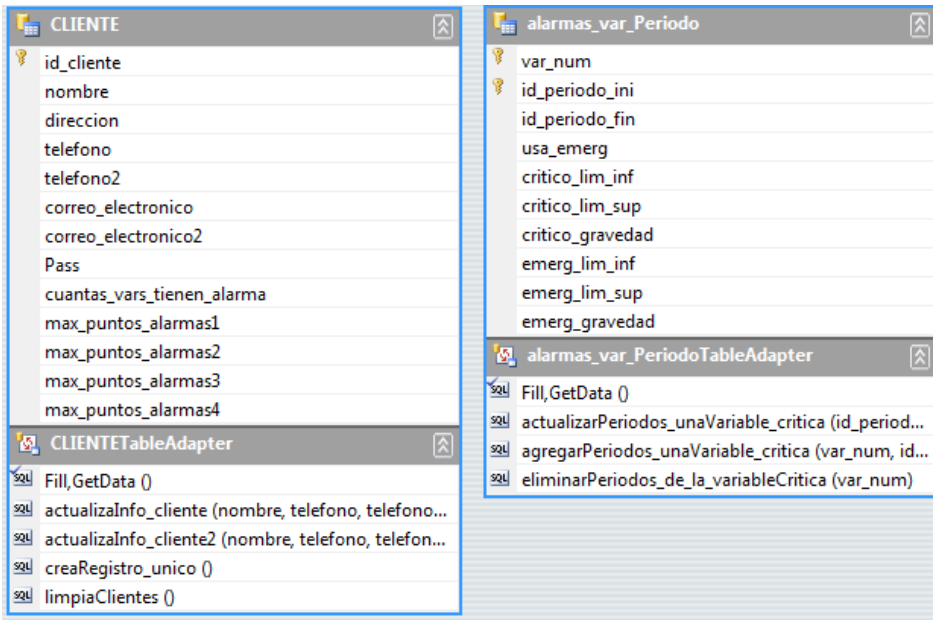


Figura 111. Bases típicas: modelo del cliente. Tablas: cliente y alarmas por periodo.

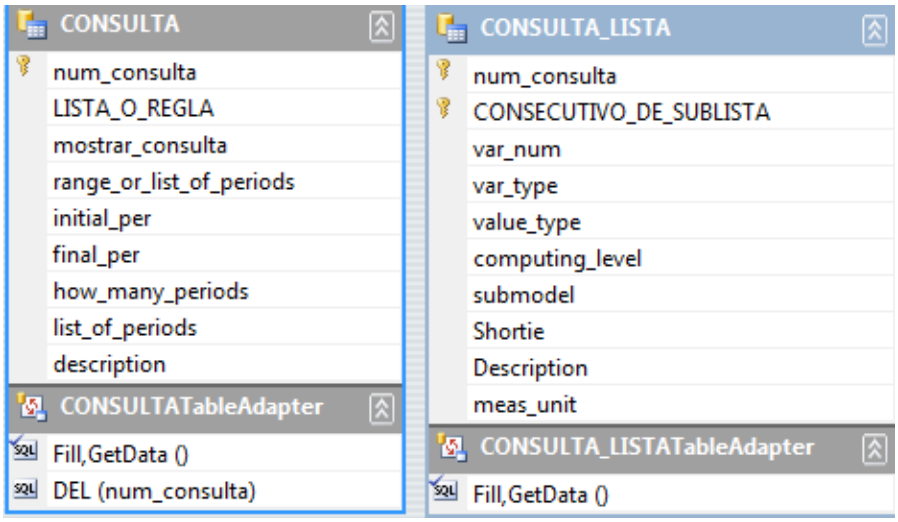


Figura 112. Bases típicas: catálogo de consultas. Tablas: consulta y consulta_lista.

El catálogo de consultas se compone de cuatro tablas. Las primeras dos, mostradas en la Figura 112, almacenan las consultas con sus datos y la lista de variables por cada consulta. En la Figura 113 se pueden ver la otras dos, una para identificar a cada regla por consulta elaborada y la otra que almacena una lista de las variables que contiene cada regla de la consulta.

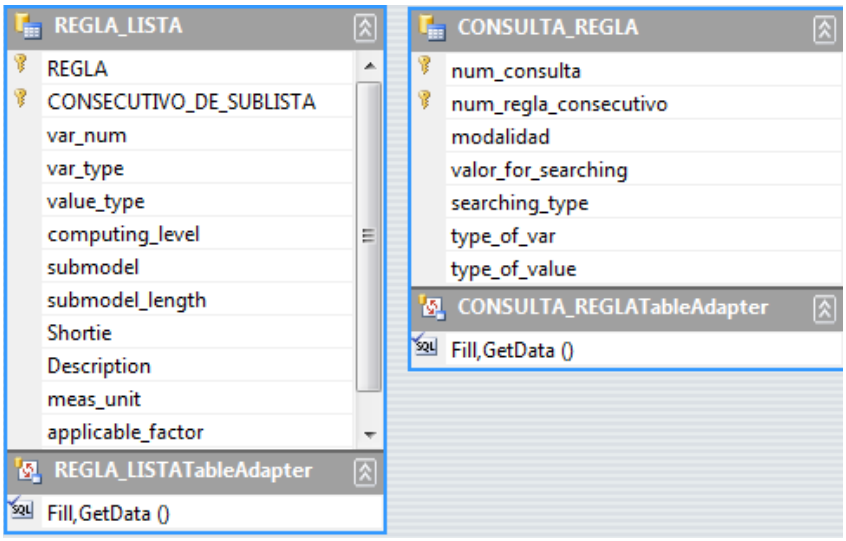


Figura 113. Bases típicas: catálogo de consultas. Tablas: regla_lista y consulta_regla.

Cabe decir que en ésta como en las demás bases de datos se hace uso de procedimientos almacenados pero generados en el diseñador que ofrece Visual Studio 2010, es decir, en el dataset donde se cargan las tablas de la base.

Por otro lado, el sistema FLAG administrador tiene dos bases principales de información:

- Modelo de las VMR. El archivo se denomina “REALVARIABLES.accbd”
- Alarmas generadas. El archivo se denomina “FLAGALARMSBASE.accbd”

El modelo de las variables del mundo real contiene en total cinco tablas:

- VMRs
- Sinónimos
- Agentes
- Cuentas de los clientes
- Criterios de ejecución de las VMR por cada cliente

Las primeras tres se muestran en la Figura 114, mientras que las restantes dos en la Figura 115. La base de datos de las alarmas generadas contiene una sola tabla denominada “alarmas generadas” como se puede ver en la Figura 116.

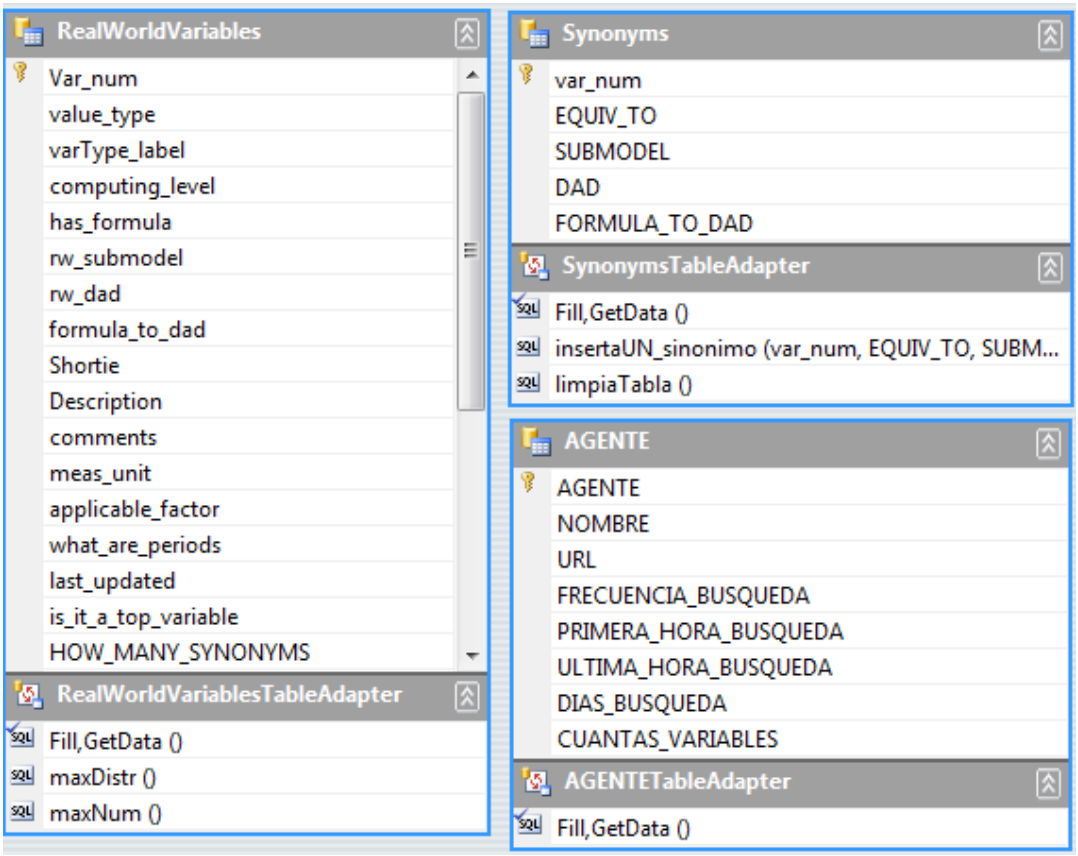


Figura 114. Bases típicas: Modelo de las VMR. Tablas: variables del mundo real, sinónimos y agentes.

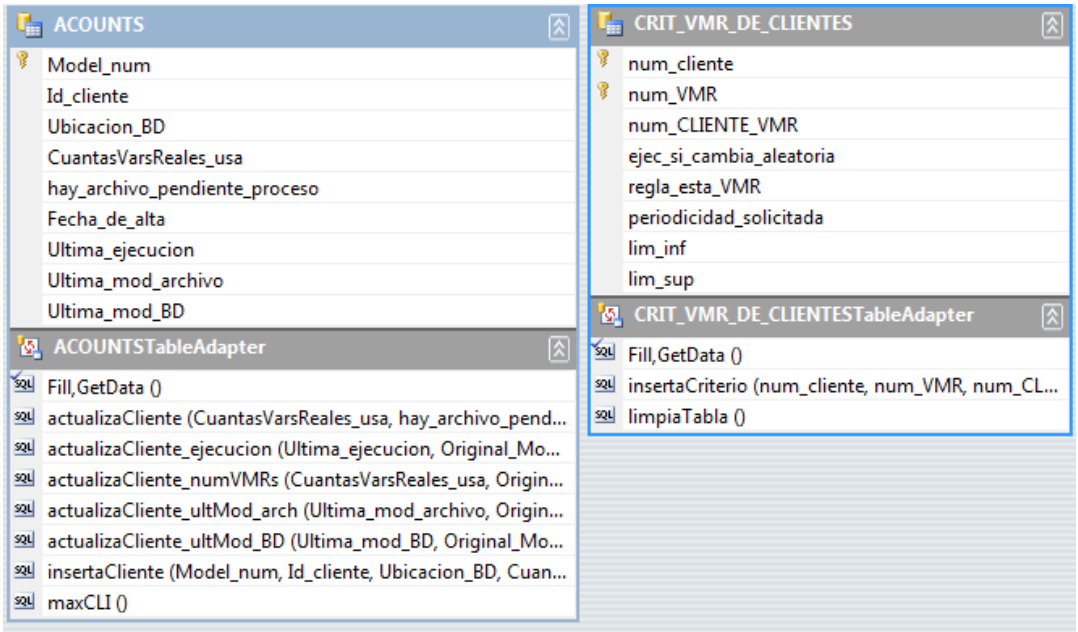


Figura 115. Bases típicas: Modelo de las VMR. Tablas: cuentas y criterios de ejecución de VMR de cada cliente

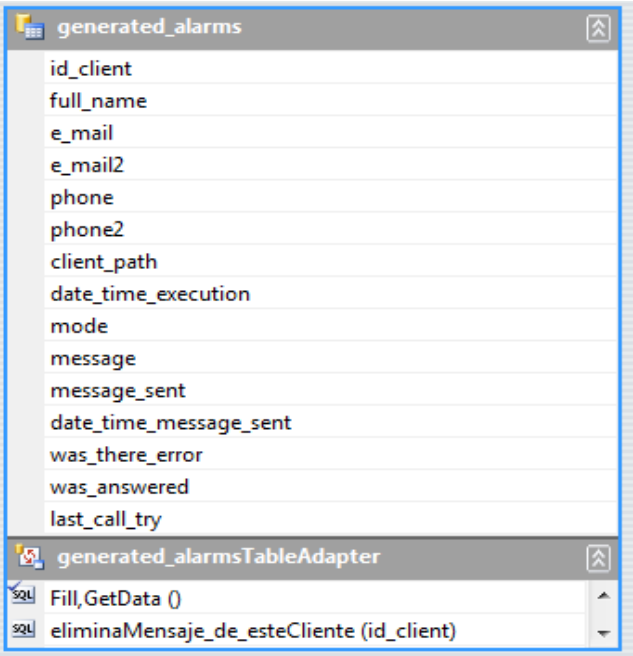


Figura 116. Bases típicas: alarmas generadas. Tabla: alarmas generadas

4. Instructivo para crear un servicio FLAG

El FLAG es un paquete de software que permite crear instancias para ofrecer los servicios a un conjunto de clientes. Para crear una instancia del FLAG, se ejecutan las siguientes actividades.

1. Se crea un directorio FLAG (en el directorio que seleccionen los interesados en ofrecer el servicio). Todos los demás datos residirán en este directorio (con excepción de los que pudieran estar en las computadoras de los clientes).
2. El directorio FLAG tiene los siguientes subdirectorios:
    a. FLAG – bases de datos
    b. CLIENTES
    c. FLAG – Archivos.
    d. FLAG – sistema
    e. FLAG – DLL
3. El que crea la instancia deberá incluir en el directorio “nuevo” estos subdirectorios, que copiará de una carpeta FLAG ejemplo.
4. Preparar los programas ejecutables en el servidor.
5. Para cada cliente del servicio:
    a. Le proporciona las bases típicas y los programas que usará;

    b. Le entregará el material de capacitación que proceda;
    c. Incluirá el nombre del directorio que le indique el cliente en su tabla de CLIENTES (en la base de datos RealVariables);
    d. Establecerá un modo de comunicación para que
        i. El cliente le envíe su base de datos y sus archivos (fórmulas y valores) cuando éstos sufran cambios;
        ii. Se le comuniquen circunstancias, adeudos, etc. al cliente.
6. Guardar los archivos en un subdirectorio de CLENTES con el nombre de la carpeta que proporciona el cliente.
7. Extraer la información de los clientes para incorporarla a los datos que usa el motor. En particular, se usan los criterios de ejecución y las frecuencias de actualización de las VMR que usa cada cliente.
8. Instalar los dispositivos necesarios para el envío de alarmas.
9. Si hay clientes que solicitan llamadas por teléfono por parte de un empleado del FLAG para comunicar una alarma, hay que preparar este servicio.
10. Para cada VMR ofrecida, conseguir la fuente de información y crear un agente que la consiga.

Comentario: como no se ha implementado el módulo “contable” (precios, facturación, cobranza) los que ofrezcan el servicio deberán definir estos aspectos y – hasta la nueva versión – manejar los cargos con otro sistema.

La versión actual sólo “contabilizará” los servicios relacionados con la ejecución de los modelos.

5. Documentos anexos en el CD

Los archivos que se incluyen en el CD, además de la tesis, se dividen en dos partes: documentos del cliente y los del sistema Administrador. Para el cliente se incluye una carpeta con lo siguiente:

- La tesis (como pdf)
- Ejecutable del sistema FLAG del cliente
- Bases de datos pobladas con información de un ejemplo con variables del cliente y variables del mundo real. Son las llamadas bases típicas.
- Archivos del sistema: fórmulas y valores
- DLLs necesarias para el funcionamiento del sistema

Para el sistema administrador se incluye lo siguiente:

- Ejecutable del sistema FLAG administrador
- Ejecutable de Agentes de información
- Ejecutable de la Ejecución de Modelos
- Bases de datos: una base vacía (llamada típica) RealVariables y otra GeneratedAlarms
- Archivos del sistema: fórmulas, valores, criterios de ejecución, reporte diagnóstico de clientes y archivo de agentes
- DLLs necesarias para el funcionamiento del sistema