CAPÍTULO 10

MODELO DE UN SOFTWARE PARA DISEÑO CONCEPTUAL CREATIVO

10 MODELO DE UN SOFTWARE PARA DISEÑO CONCEPTUAL CREATIVO

10.1 Introducción

En este capítulo se presenta la propuesta de un software de creatividad orientado específicamente a la asistencia del ingeniero de diseño en la fase de conceptualización de nuevos productos. El fundamento de la propuesta lo constituyen los resultados obtenidos en la fase experimental, es decir, el estudio cuantitativo y cualitativo del efecto de diferentes técnicas de creatividad en la generación de ideas para la solución de un problema de diseño. Para ello, resulta imprescindible seleccionar previamente un modelo de representación del proceso de diseño conceptual, que se constituya en el marco orientador para el ingeniero de diseño. Por esa razón, en la primera parte del capítulo se presenta el modelo seleccionado, detallando en las características que lo definen y sus efectos en la construcción del software.

En la segunda parte del capítulo se presenta la estructura básica del programa construida sobre la base del modelo previamente comentado. Para ello se detallas las tres etapas en que se ha subdividido el proceso. En primer lugar la construcción del modelo funcional del problema, luego la búsqueda de una solución ideal y finalmente la evolución funcional hacia una solución real.

La conceptualización de soluciones a problema de diseño viene estrechamente ligada con elementos del conocimiento y de la experiencia del ingeniero. La evidencia experimental señala la conveniencia de incluir en el software propuesto una serie de herramientas que asistan al usuario en la recuperación de información y conocimiento. Por ello en la tercera parte del capítulo se hace una descripción de las bases de datos que deberían formar parte del programa.

Es claro, entonces, que con base en los criterios definidos en estas tres partes, se puede caracterizar la estructura del software deseado. Sin embargo, será necesario complementar la propuesta introduciendo algunos elementos importantes relacionados con la usabilidad de software. El experimento realizado muestra que los programas evaluados tienen serias limitaciones en este aspecto y será conveniente tener en cuenta que la estructura del software por muy bien definida que esté, carecerá de utilidad si no está acompañada de elementos claros de interacción con el usuario. Por ello, en la cuarta parte de este capítulo se recogen algunas pautas a seguir en la formalización del software.

Como un complemento a la propuesta se presenta el tema de la esquematización gráfica (en forma de dibujos), como un elemento de alta relevancia en el proceso de conceptualización en diseño. Allí se enseñan las alternativas que podrían llegar a ser solución a este inconveniente de las aplicaciones informáticas.

Se concluye esta investigación en la última parte del capítulo donde se recoge todas las definiciones dadas anteriormente para elaborar una "maqueta" de software, esto es, un modelo de la forma que deberá tener el programa. Como tal, la maqueta aunque carece de funcionalidad muestra la forma deseable del programa con todos sus componentes, dando así las bases para el futuro desarrollo y codificación del software.

10.2 Modelo teórico estructural del software

Revisados y expuestos los argumentos para utilizar el modelo FBS (Umeda, 1990) para representar el proceso de diseño en el capítulo 7, en esta sección se precisarán algunas de sus características particulares que permiten seleccionarlo como el modelo que da la estructura teórica al software que se quiere proponer. Aunque la literatura reporta su uso para propósito de análisis de resultados experimentales de procesos ya ejecutados, aquí se propone su utilización para estructurar el proceso antes de ejecutarse, es decir, se le dará una connotación de tipo prescriptivo.

En primer lugar cabe recordar que el elemento alrededor del cual gira tal modelo es la *función*, desempeñando sus tres roles importantes en diseño: medio de modelación de los requerimientos del cliente (especificaciones) y de la evolución del proceso, medio de articulación entre los requerimientos y el objeto, e instrumento de evaluación del valor del objeto diseñado. La estructura de la función, como la define Takeda (1996), está representada como una combinación del cuerpo de la función o función, a secas (verbos), la entidad objetivo o estructura física que desarrolla la función (sujeto), y los modificadores funcionales, que califican la función (adjetivos y/o adverbios).

La evolución funcional representa los diferentes estados por los que pasa el proceso de diseño y está caracterizada por las relaciones entre funciones y entre éstas y los modificadores funcionales que se dan a través del tiempo. Esos diferentes estados evidencian las etapas convergentes y divergentes que se dan durante el proceso, que Liu et. al. (2003) describen como etapas múltiples divergentes y convergentes, ilustradas en la Figura 10.1.

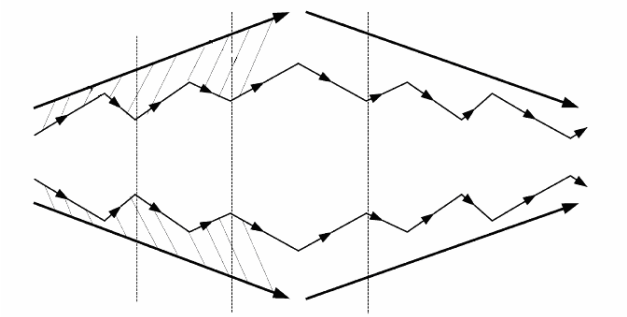

Figura 10.1 Etapas múltiples divergentes-convergentes en el proceso de diseño conceptual
Fuente: Liu et. al (2003)

En este modelo, el diseño conceptual sigue dos etapas básicas: una primera con una tendencia divergente en la búsqueda de alternativas de solución y la segunda, convergente, que evalúa las alternativas para centrarse en las más prometedoras. Sin embargo, como lo ilustra la gráfica, en cada etapa se darán procesos múltiples de convergencia-divergencia, lo cual representa más fidedignamente el proceso real y puede resultar más eficiente para un entorno computacional (Mulet, Vidal y Gómez-Senent, 2003).

Los ciclos de divergencia estarán marcados por la generación de funciones y sus subdivisiones, que de acuerdo con Takeda (1996) son de tres tipos:

- Descomposición, cuando se divide una función en sub-funciones. Por ejemplo, la función «plegar empaques» se puede descomponer en «doblar pestañas» y «estirar empaque».
- Causalidad, cuando se identifica una función B requerida para que la función A pueda llevarse a cabo. Por ejemplo, para cumplir la función «estirar empaque» puede identificarse la necesidad de «sacar el aire del interior del empaque».
- Reforzamiento, cuando una función B se recomienda para que la función A sea más adecuada. Así, podría recomendarse la función «desprender boquilla del empaque» para que «sacar el aire del interior del empaque» se realice más fácilmente.

La etapa de divergencia de cada ciclo esta caracterizada por el surgimiento de ideas que sean soluciones potenciales para las funciones que se están analizando. Obviamente la creatividad juega aquí su papel más relevante y el software debe, precisamente, ayudar al usuario a desarrollarla con efectividad. Mulet, Vidal y Gómez-Senent (2003) sugieren que la cantidad de funciones que se analizan cada vez y sobre las cuales se desarrolla cada etapa divergente, sea reducida, con el fin de garantizar una rápida convergencia,

antes de pasar a estudiar otras funciones, continuando así el ciclo divergente-convergente.

Por otro lado, la convergencia estará asociada con la evaluación de las estructuras (objetos) que se van generando para desarrollar las funciones, realizada básicamente a través de la calificación de los modificadores funcionales. Es decir, la convergencia ocurre cuando se encuentran estructuras que satisfacen los requerimientos descritos por medio de los modificadores funcionales y se descartan las que no lo hacen.

La Figura 10.2 muestra un ejemplo del diagrama típico construido para representar el modelo FBS. En él, las funciones se muestran como circunferencias y se describen como verbos. La relación entre funciones se representa por las flechas y se identifica su clase con las letras C, D y R (causalidad, descomposición y reforzamiento). Los modificadores funcionales se identifican con la letra M inicial y se describe con adverbios y con adjetivos. Las estructuras son sustantivos y se representan con un rectángulo.

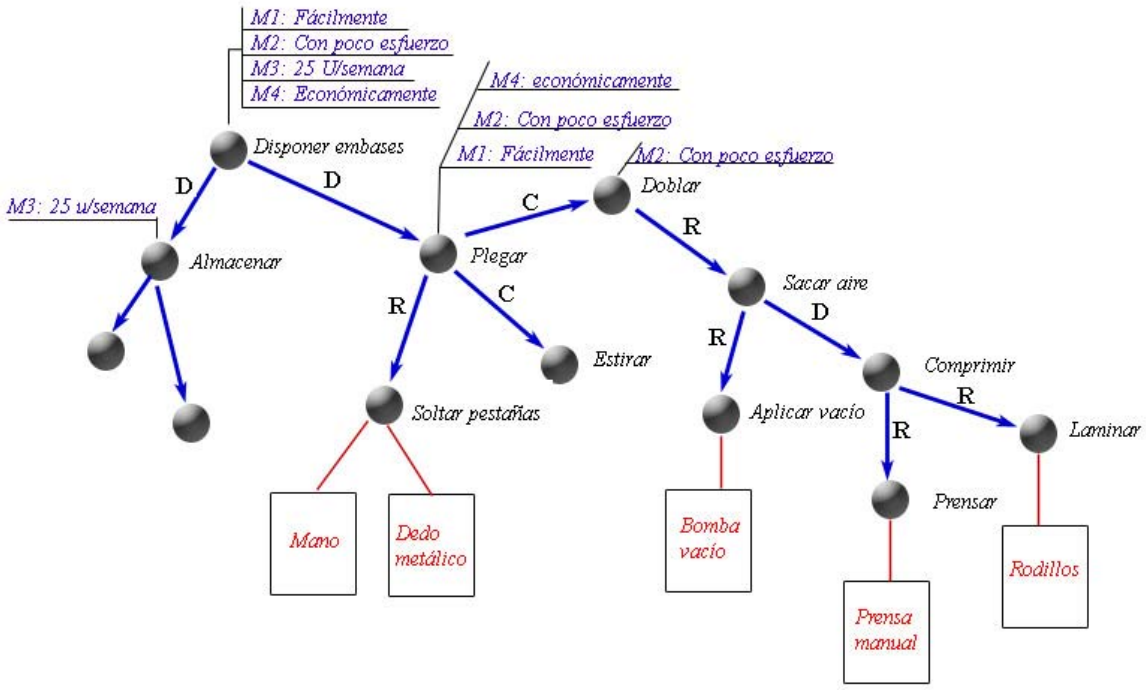

Figura 10.2 Ejemplo de representación de evolución funcional

Una de las conclusiones más importantes de la fase experimental es la necesidad de que el software mantenga un equilibrio entre la estructura y la libertad de acción, esto es, entre una forma que resulte familiar al ingeniero quien normalmente tiene una estructura de pensamiento racional y sistemática, pero que a la vez le deje la libertad de expresar

sus ideas a voluntad. La propuesta de guiar el proceso utilizando un modelo como el FBS, trasmitirá un sentido de seguridad al usuario, es decir, un sentido de que se transita por un camino que conduce a la solución, pero a la vez dará la libertad suficiente para que la construcción de tal solución y la solución en sí misma, sean creativas.

Se propone que la interacción en el software sea de tipo gráfico, con cuatro elementos básicos: uno que represente las funciones y sub-funciones, otro para representar las estructuras de solución, un tercero que represente los modificadores funcionales y un cuarto para representar las interacciones entre elementos anteriores. Su forma de visualización será por niveles (capas): un primer nivel que muestra las funciones y sub-funciones, un segundo nivel que corresponderá a los modificadores funcionales y un tercer nivel que es correspondiente a las estructuras de solución.

10.3 Etapas del proceso

El software guiará al usuario a desarrollar el proceso en tres etapas que en un principio se presentan como secuenciales, pero que permiten retroalimentación, con lo cual es posible regresar a cualquiera de ellas en cualquier momento. La Figura 10.3 muestra esquemáticamente las tres etapas, que se describen a continuación.

10.3.1 Primera etapa: Modelo funcional del problema a resolver

En esta etapa se hace la interpretación funcional de los requerimientos y especificaciones del problema de diseño. Se parte del supuesto de que el problema ya se ha definido y se han identificado las especificaciones y requerimientos que debe cumplir una solución para su aceptación. Esta etapa previa podría ser realizada en un módulo preliminar que integre herramientas como el QFD. Para ello existen dos opciones: integrar el programa con alguno de los ofrecidos comercialmente o desarrollar un modulo propio.

Aquí no se presenta la discusión de estas alternativas porque se sale de los objetivos propios de la investigación, sino que se asume como elemento de partida la definición del problema de diseño, en términos de funciones deseadas por el cliente o usuario. Esta forma de definición será soportada por el software por medio de su interfase gráfica y con la ayuda de una base de datos de funciones que ayuden a formularlas gramaticalmente con precisión como actividades, acciones, servicios a prestar o necesidades a satisfacer.

Tassinari (1994) cita la definición de función aceptada por la norma francesa NX 50-150, como «las acciones de un producto o de uno de sus componentes expresada

exclusivamente en términos de su finalidad» (pp.37-38), es decir, en términos de objetivos funcionales.

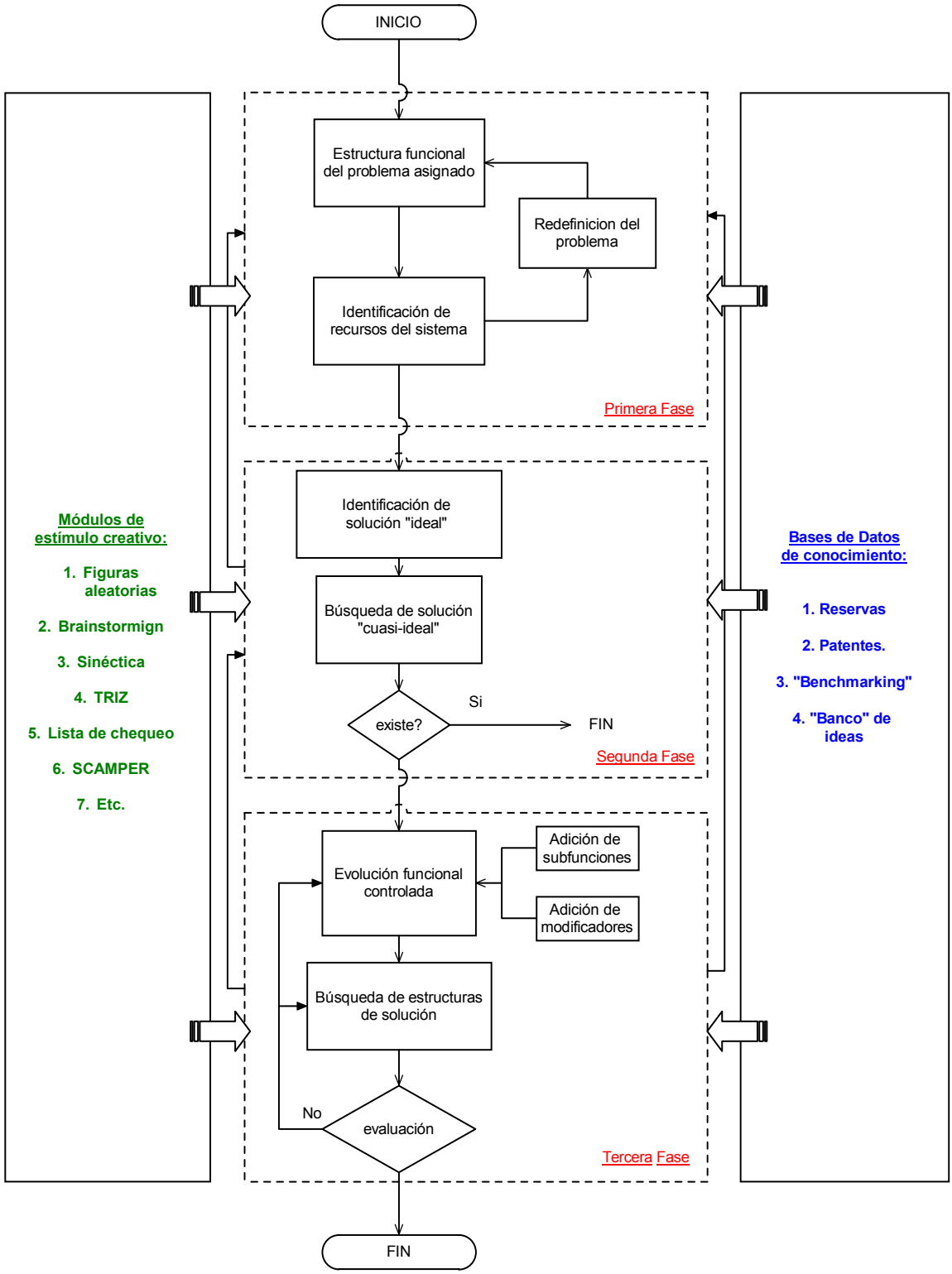

Figura 10.3 Diagrama del proceso de diseño conceptual asistido por el software propuesto.

Definido el problema en términos de funciones deseadas, en el programa se procede a describirlo gráficamente. Un ejemplo del modelado gráfico del problema asignado se muestra en la Figura 10.4 en el que se aprecian dos funciones y cinco características de ellas (denominadas, modificadores funcionales).

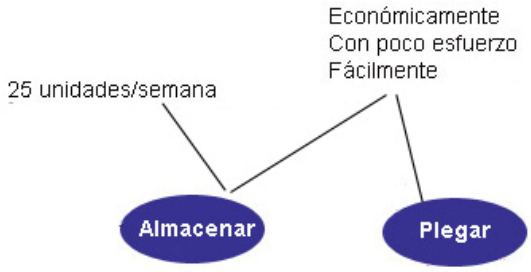

Figura 10.4 Modelo funcional del enunciado del problema

Como ya se comentó, los modificadores son básicamente criterios de valoración de las funciones, que entregan elementos para la convergencia del proceso. Evidentemente el ejemplo mostrado corresponde a un caso simple de problema de diseño, en el que los criterios son amplios y un tanto ambiguos. En la práctica real, tales modificadores deberían ser más precisos y preferiblemente cuantificables, eso sí sin tener que llegar a una identificación tan exhaustiva como la requerida, por ejemplo, cuando se aplica la técnica del análisis del valor a un producto. Para ello el software debe dar una orientación que facilite la correcta y completa expresión de modificadores, principalmente a través de una base de datos de conocimiento de la propia empresa, que se alimenta con los datos de proyectos ya ejecutados, almacenados ordenadamente y que conforma lo que se podría llamar «memoria técnica» de la empresa y que será presentada con detalle más adelante.

En esta fase también se realiza un refinamiento del modelo funcional del problema asignado, utilizando dos de las herramientas presentes en el software y que tiene su fundamento en la metodología TRIZ.

La primera tiene que ver con la identificación de los «recursos del sistema». Aquí, sistema se entiende como el conjunto de elementos y sus interrelaciones que constituyen el contexto o el entorno del problema a resolver. Se proponen cuatro grupos diferentes de recursos: sustancias y materiales, energías, reservas y funciones adicionales. En el primero quedan comprendidas todas las sustancias sólidas, líquidas y gaseosas (incluyendo desechos o residuos), disponibles en el sistema o en su ambiente, tanto en su estado original como en formas derivadas o modificadas. En el segundo grupo, todas las formas de energía disponibles o aplicables al sistema, es decir, cualquier clase de

influencia que el objeto pueda ejercer o se le puede ejercer (mecánica, térmica, química, eléctrica o magnética). El tercer grupo está comprendido por las reservas no utilizadas de: tiempo libre, espacio no ocupado e información adicional. Y en el cuarto grupo se identifican las funciones adicionales que los elementos del sistema podrían desempeñar.

El software, por lo tanto, potenciará la identificación de recursos del sistema para que el usuario tenga una visión completa de todos los elementos que podría utilizar para la solución del problema de diseño. Para ello existirá una opción que despliegue una «lista de chequeo» alimentada por una base de datos de recursos según las categorías antes comentadas, como la mostrada en la Figura 10.5.

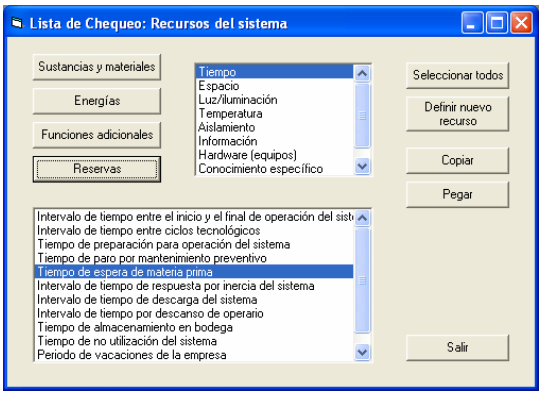

Figura 10.5 Cuadro de control con la lista de chequeo de recursos del sistema

La identificación de recursos se constituye en un paso importante para contrarrestar lo que se ha denominado como «inercia mental», esto es, la tendencia que tiene el ingeniero a pensar soluciones únicamente dentro de su campo de conocimiento, sin explorar posibilidades en otras áreas.

La segunda herramienta disponible para la refinación del modelo funcional del problema tiene que ver con un cambio de perspectiva de la situación, lo que puede llevar a hacer una «redefinición del problema». Para ello se utilizará una técnica muy simple pero que permite la exploración a profundidad las causas de un problema, propuesta inicialmente por Min Basadur, citado por Apte, Shan y Mann (2000) e incorporada en el software CREAX, que consiste en formular reiteradamente la pregunta «por qué quiero solucionar el problema?», para ampliar la visión del problema, y la pregunta «qué me impide solucionar el problema?», para estrechar la visión del problema. Aquellos cuestionamientos llevan a cambiar la escala de análisis del problema, y es probable que se modifique el enfoque del proceso de solución, como se ilustra en la Figura 10.6. Como resultado se obtendrá una lista jerarquizada de redefiniciones del problema y el usuario podrá seleccionar el que mejor convenga.

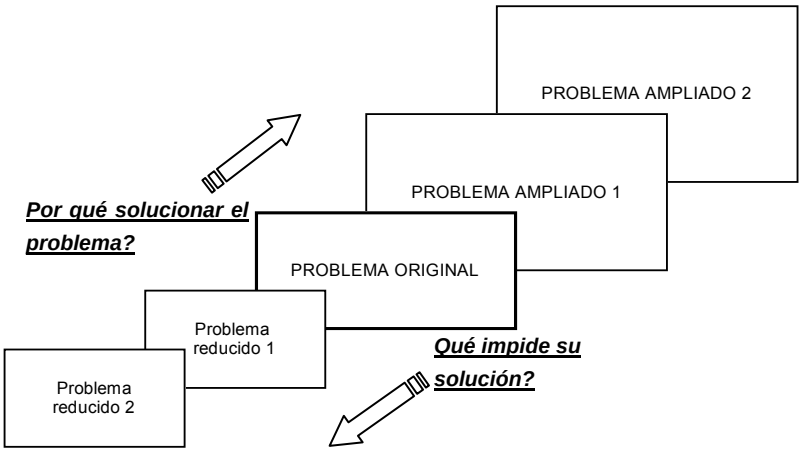

Figura 10.6 Redefinición del problema a resolver por cambio de enfoque.

Como el elemento básico del proceso es la función, la redefinición se aplicaría al modelo funcional del problema asignado. La Figura 10.7 muestra un ejemplo de la apariencia que tendría la interfase para esta etapa del proceso.

Figura 10.7 Modelo de la interfase en el proceso de redefinición del problema

Cuando el usuario invoca la herramienta de redefinición, se abrirá una ventana conteniendo el modelo funcional del problema asignado pero sin los modificadores

funcionales. Las funciones aparecerán centradas en la interfase y las herramientas disponibles permitirán avanzar hacia arriba mediante la respuestas a la pregunta reiterada de «por qué?» y hacia abajo por la pregunta «que impide?» Cada respuesta dada deberá reinterpretarse en forma de función. En el ejemplo de la figura, la primera pregunta se leería: «por qué se debe plegar el empaque?», y la respuesta ha sido: «porque el empaque ocupa mucho espacio». Tal respuesta se interpreta funcionalmente como: «reducir el volumen del empaque vacío», generando de esta manera un nivel más amplio del problema asignado, de manera que el diseñador podría percibir el problema bajo una perspectiva diferente y con ello adoptar un objetivo diferente para el producto a desarrollar.

De esta manera se termina la primera etapa del proceso. Esto no significa, como ya se comentó, que el resultado final de esta etapa sea rígido y no se pueda modificar posteriormente. Será solamente el punto de partida con todos los elementos necesarios para avanzar hacia una solución, proceso durante el cual podría surgir la necesidad de modificar el enfoque y por lo tanto cambiar el modelo funcional original.

10.3.2 Segunda etapa: Búsqueda de la idealidad

Cuando se aborda esta etapa ya se ha definido el problema a resolver y se han identificado los recursos disponibles del sistema que lo enmarca. Se trata ahora de abordar el reto de encontrar una solución rápida e ideal, aprovechando tales recursos. En general se puede decir que existe una tendencia del ingeniero a mirar siempre soluciones complejas a los problemas y a no aceptar la posibilidad de una solución sencilla. Por ello, antes de emprender una búsqueda más elaborada y compleja, se presenta al usuario la posibilidad de pensar en otro tipo de soluciones y este es el objetivo de la segunda etapa propuesta en el software.

La idealidad tal como se entiende en la metodología TRIZ se define como la relación entre la suma de las funciones o efectos deseados y la suma de las funciones o efectos indeseados (Terninki, Zuzman y Zlotin, 1998), lo que equivale al concepto de «valor» en la metodología de análisis de valor:

$$Idealidad \ \ (valor) = \frac{\sum efectos \ deseados}{\sum efectos \ indeseados + \sum costes} \qquad (10.1)$$

Dentro de los efectos no deseados se incluyen aspectos tales como costes, espacio ocupado, ruido que produce, energía que consume, tiempos de retardo, efluentes

emitidos, etc. Por ello, entre mejor se cumpla una función deseada (incremento de efectos deseados) o menos efectos indeseados haya, se tendrá una mejor solución.

Una de las leyes más importantes de la teoría del desarrollo de los sistemas técnicos deducidas por Altshuler (1988) después de estudiar los patrones seguidos por un gran número de patentes, es la ley del «incremento de la idealidad», la cual establece que los sistemas técnicos tienden desarrollar mejor sus funciones y a disminuir sus efectos indeseados, siendo la máxima expresión de idealidad, llamada por Altshuler «RFI: resultado final ideal» (Kaplan, 1996, p.4), aquella que se logra cuando se obtienen todos los efectos deseados y ningún efecto indeseado. Esto significa que se logra cumplir la función deseada por sí misma, sin ningún elemento o artefacto artificial al sistema. Un ejemplo que permite aclarar este concepto de idealidad es el siguiente:

> *Se requirió comparar la resistencia de diferentes aleaciones metálicas al ataque de un ácido. Originalmente se colocaron varias probetas durante un tiempo determinado dentro de un recipiente lleno del ácido, para luego retirarlas y evaluar el efecto. El problema de este procedimiento es que el ácido también atacó las paredes del contenedor, por lo cual se pensó en recubrirlo con algún material resistente al ácido. Sin embargo está solución resultaba muy costosa. Así se aplicó la ley de la idealidad, es decir, se buscó que se cumpliese la función de mantener en contacto la probeta y el ácido sin necesidad de contenedor. La solución fue elaborar la probeta con forma de recipiente y verter dentro el ácido. Así, se eliminó la necesidad de un contenedor y se logró el efecto del ácido sobre la probeta* (Terninko, Zuzman y Zlotin, 1998, p.95-96).

Gráficamente se puede entender el principio como lo muestra la Figura 10.8. En ella se plantea una situación actual (problema a resolver) y una situación ideal en el otro extremo. Normalmente se desarrolla la tendencia de ir introduciendo mejoras a la situación actual de manera que se va produciendo una evolución gradual hacia estadios superiores de la tecnología. Si el objetivo se centra en la situación ideal (la máxima expresión tecnológica para la solución del problema) se tendrá una mejor posición para dar el “salto” tecnológico y entonces, si es necesario, es decir, si no se logra obtener el RFI, se darán pasos hacia atrás y en todo caso, se estará asegurando la mejor solución tecnológica viable actualmente. El tener como meta el resultado final ideal, estimulará el pensamiento hacia soluciones verdaderamente innovadoras, ya que se darán “saltos” en la secuencia evolutiva de la tecnología, es decir, se evitará pensar solamente en mejorar un producto ya existente (innovación incremental) y más bien se potenciará la búsqueda del ideal.

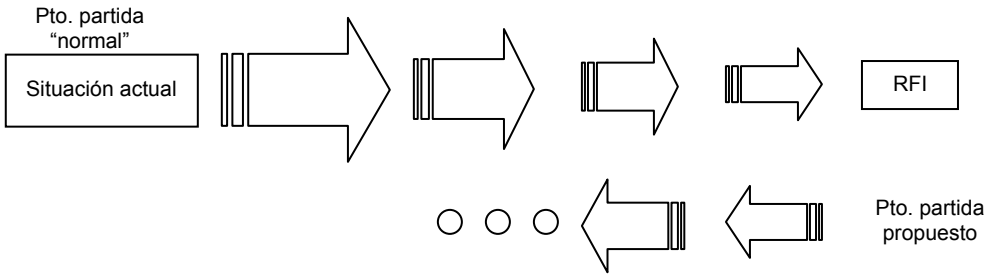

Figura 10.8 Proceso de evolución hacia el resultado final ideal, RFI

La definición del RFI es una tarea que no siempre es sencilla, pero que debe abordarse si se quiere llegar allí. Para ello conviene valerse de los resultados obtenidos en el proceso de redefinición realizado en la fase anterior. Aun cuando no se haya modificado el problema a resolver, la posibilidad de verlo desde perspectivas más amplias o mas estrechas posibilita identificar el RFI. El software, por lo tanto, mostrará la gráfica de redefinición previamente elaborada y permitirá la edición textual/gráfica del RFI.

Según Domb (1997) el RFI describe la solución a un problema técnico independientemente del mecanismo o las restricciones del problema original. No ocupa espacio, no tiene peso, no requiere trabajo, ni mantenimiento. El RFI permite obtener la función requerida sin desventajas y sin costes. El RFI tiene las siguientes cuatro características:

- Elimina las deficiencias del sistema original.
- Preserva las ventajas del sistema original.
- No hace más complicado el sistema original.
- No introduce nuevas desventajas.

Generalmente el RFI se constituye en un ideal inalcanzable, una utopía. Sin embargo, se puede percibir como una herramienta psicológica que orienta el uso de las herramientas técnicas. Formular el RFI ayuda a mirar las restricciones del problema y a considerar cuales de ellas corresponden a limitaciones de las leyes de la naturaleza y cuales son artificiales (auto-impuestas). De esta manera se identifican las fronteras reales de la solución, para buscar una solución a partir de tales fronteras.

Existen varias alternativas para la búsqueda de la solución ideal: utilizar recursos del sistema, eliminar funciones auxiliares, excluir o remplazar elementos del sistema (o

incluso el sistema en su totalidad), identificar auto-servicios y cambiar los principios de operación.

Evidentemente, el utilizar recursos disponibles en el sistema se constituye en una herramienta aplicable especialmente cuando se parte de un análisis funcional como el que aquí se propone y cuando el objetivo es lograr una solución nueva. Cuando se parte de un producto ya existente que se quiere mejorar o remplazar, también cobran vigencia las otras herramientas mencionadas.

El software debe incluir entonces la posibilidad de aplicar tales herramientas en la búsqueda de una solución ideal. Una primera aproximación consiste en producir frases que combinen los recursos del sistema (que ya fueron identificados en la fase anterior) con las funciones del problema (también identificadas en la modelo funcional), en forma semejante a como lo hace el programa Axon Idea Processor, como se muestra a modo de ejemplo en la Figura 10.8. Tales frases o preguntas ayudarán al usuario a encontrar posibles respuestas que sean muy creativas para resolver el problema.

En el ejemplo se muestra la combinación de la función principal «plegar» con los recursos «sustancias y materiales» (aire, muebles de cocina, frigorífico, agua, gas natural, etc.), asumiendo que estos fueron los identificados en la primera fase. Pero existen otras muchas posibles combinaciones de estas palabras que ayudarían a generar ideas creativas. Preguntarse si alguno de los recursos podría cumplir funciones auxiliares de sistema, o si se podría delegar en ellos funciones que cumplen algunos elementos del sistema. Evidentemente estas posibilidades de combinación tendrán mas aplicación cuando se este trabajando en la siguiente fase (evolución funcional controlada) explicada posteriormente.

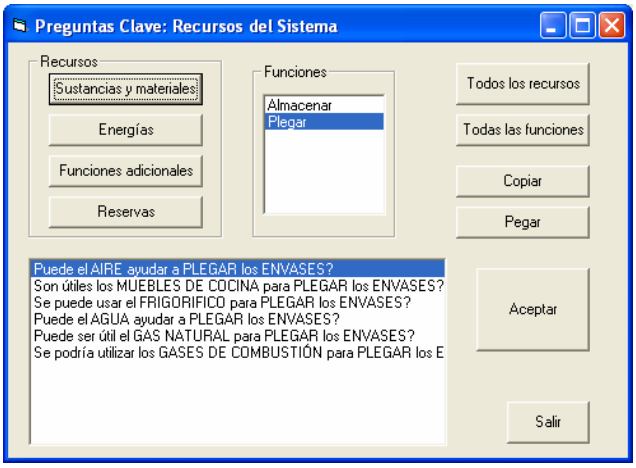

Figura 10.9 Cuadro de ayuda para buscar solución ideal mediante uso de recursos del sistema

Otra forma de utilizar los recursos disponibles en el sistema como fuente de soluciones es explorar la posibilidad de utilizarlos previa transformación o modificación. Así, podría preguntarse cosas como: ¿qué pasaría si en lugar de utilizar el agua líquida utilizara hielo?; podría utilizar alguna reacción química del butano para solucionar el problema?. El espectro de uso de recursos se amplía en gran manera al pensarlos como elementos transformados o modificados, aunque esto implique la necesidad de mayor conocimiento para poder explorar áreas muy diversas.

Será conveniente para estos efectos, disponer en la base de datos del software de información relacionada con procesos físicos, químicos, mecánicos, etc. que sirvan para transformación de recursos y que el usuario las pueda consultar fácilmente. Una aproximación del contenido y estructura de esta base se presentará en el tema de las bases de patentes.

Además de la base de datos mencionada, la identificación e incorporación de nuevos recursos estará asistida por una de las herramientas más útiles durante la fase experimental, esta es, la presentación de figuras aleatorias. La información presentada en forma gráfica facilita una rápida interconexión de conceptos que puede originar nuevas ideas, en este caso, orientadas a la identificación de recursos del sistema que pueden haberse pasado por alto o a formas creativas de utilización de tales recursos que permitan obtener el resultado final ideal.

La búsqueda de principios físicos y principios de trabajo constituye otra fuente importante de ideas de solución, principalmente porque es una fuente de información que ayuda a eliminar los bloqueos que se presentan por desconocimiento en un área determinada de la técnica. Es normal la tendencia que se tiene a centrarse en soluciones que estén en el dominio del conocimiento específico del usuario; por ejemplo, si es un ingeniero mecánico, buscará soluciones de tipo mecánico, eludiendo o ignorando posibles soluciones en otras áreas, como la electrónica o la química, que podrían dar luces sobre mejores soluciones que las puramente mecánicas.

Se entiende que el trabajo en equipos multidisciplinarios busca este enfoque amplio de soluciones y por ello la tendencia moderna de trabajar de esta manera. Sin embargo, la eficiencia del equipo estará supeditada a la capacidad que tengan sus miembros de comunicarse adecuadamente, esto es, de tener un mínimo manejo conceptual de varias disciplinas. Sería entonces de gran ayuda que el software pudiese ofrecer información sobre principios físicos de varias disciplinas presentados de tal manera que sean fácilmente comprensibles, con un lenguaje simple y que tenga la posibilidad de enlaces con fuentes de información que permitan al usuario profundizar si lo considera necesario.

Evidentemente incorporar directamente esta información en el software no resultaría muy eficiente, por la cantidad de información requerida y porque ya existen muchas fuentes de información disponibles en la red de Internet. Convendría por lo tanto que el software dispusiera de una herramienta que muestre un índice organizado de acuerdo con alguno de los sistemas de clasificación, por ejemplo el sistema UNESCO (Universidad del País Vasco, 2004) y permita el enlace a alguna de las enciclopedias técnicas especializadas de libre acceso como por ejemplo la Enciclopedia libre universal en Español (2004), la Wikipedia (Wikipedia Foundation Inc., 2004) o la Ciberoteca (Bancaza, 2001). Los usuarios, entonces, podrían consultar información relevante para el problema planteado en varias disciplinas, ampliando de esta manera las posibilidades de aproximarse a la solución ideal.

Un método adicional que puede utilizarse para los efectos de la búsqueda del resultado final ideal es el utilizado por la sinéctica y que tiene que ver con la empatía: el juego de rol. Consiste en asumir la perspectiva que tendrían diferentes personajes para abordar el problema. Los programas ThoughtPath y Braisntorming lo implementan proponiendo al usuario asumir diferentes roles, que se pueden seleccionar a voluntad o permitir que el programa los proponga de manera aleatoria. Los detalles de esta técnica son presentados en el siguiente numeral, junto con las otras técnicas de creatividad.

10.3.3 Tercera etapa: Evolución funcional controlada

Finalizada la búsqueda del resultado final ideal y en caso de no lograrse obtener, se pasará a la tercera etapa del proceso, denominada evolución funcional controlada, debido a que se desarrollará la búsqueda de soluciones mediante la evolución funcional, es decir, el despliegue controlado de sub-funciones o funciones derivadas ya sea por descomposición, por causalidad o por reforzamiento. Cada vez que se agregue una sub-función o grupo de sub-funciones relacionadas por un objetivo común, el usuario es estimulado a la búsqueda de estructuras de solución y a su evaluación, con lo cual se ejercerá un control del proceso forzando el desarrollo de ciclos completos y alternativos de divergencia y convergencia.

Al entrar en esta etapa el usuario ya tendrá claro tanto el problema a resolver (primera etapa del software) como el resultado final ideal (segunda etapa) y las razones que han impedido la adopción de tal solución. En otras palabras, el usuario tendrá la visión del punto de partida y del punto de llegada, y buscará entre aquellos dos, una solución intermedia, pero siempre avanzando de lo «más ideal» a lo «menos ideal», tal como se enseña en la Figura 10.10 donde cada flecha azul representa uno de los ciclos completos de la evolución funcional.

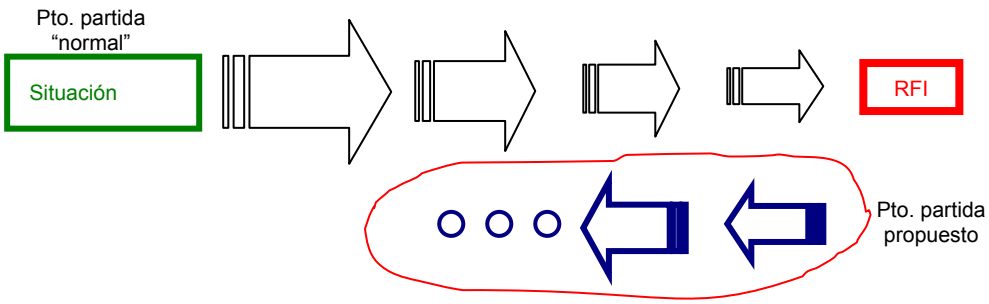

Figura 10.10 Representación del proceso de evolución funcional controlada

Durante el ciclo de divergencia, cuando se proponen nuevas funciones y se buscan estructuras de solución, el software ofrecerá al usuario las bases de datos de conocimiento, así como las diversas herramientas de creatividad de que dispone y que serán presentadas en detalle en la siguiente sección. Las bases de datos tendrán protagonismo principalmente en la definición de las sub-funciones y de sus modificadores asociados, mientras que en la búsqueda de estructuras de solución entrarán a jugar papel importante las herramientas de creatividad. Esto no significa que estos dos tipos de herramientas sean excluyentes entre sí, sino que buscan ser complementarias.

La interfase deberá mostrar resaltada la función que se está estudiando en un momento determinado, pero sin dejar de enseñar el conjunto de funciones de todo el sistema o subsistema (cuando haya necesidad de dividirlo). Con ello se busca que la atención se centre en aquella función, pero sin perder la visión de conjunto. Este tipo de interfase se propone teniendo en cuenta que uno de los principales inconvenientes del software estudiado es la polarización que muestra en la estructuración de la solución. Algunos son tan flexibles que hacen que el usuario pierda el objetivo, mientras que otros son tan estructurados que terminan por ahogar el proceso creativo.

10.4 Módulos de estímulo creativo

En función de los resultados obtenidos por la evaluación experimental, se presentan a continuación los módulos y herramientas más significativas de cada uno de los programas, así como las características que fueron destacables en cada caso. Con ello se pretende cubrir un amplio espectro y definir las opciones que deben estar disponibles en un software creativo orientado a la asistencia en el proceso de conceptualización de nuevos productos.

Algunas de las técnicas que produjeron mejores resultados ya se han presentado como parte integrante de la estructura del software: los mapas mentales (Axon Idea Processor) que se usan en la representación del modelo funcional, los recursos, redefinición y relator (Creax Innovation Suite) ampliamente tratados en el tema de la definición y búsqueda del resultado final ideal.

Otras técnicas que resultaron significativas en cada uno de los programas estudiados y que se utilizaran en este programa con las variantes oportunas, son:

- Brainstorming toolbox: figuras y palabras aleatorias, SCAMPER y juegos de rol.
- Axon Idea Processor: preguntas.
- Creax Innovation Suite: principios inventivos y contradicciones técnicas.

10.4.1 Figuras aleatorias

El estudio experimental ha revelado que el mayor número de ideas fue estimulado por este tipo de ayuda. El establecimiento de relaciones e interconexiones se facilita mediante la presentación gráfica, debido a que transmite de un solo vistazo mucha más información que la palabra escrita.

Un inconveniente que se encontró en este tipo de estímulo fue la escasez de opciones. La base de datos del programa (Brainstorming toolbox) contenía cerca de 80 figuras, con lo cual era frecuente la aparición de figuras repetidas. Otra observación realizada por los participantes fue la conveniencia de cambiar de figura cada cierto tiempo en forma automática. Será necesario por lo tanto, incluir alguna posibilidad de renovación automática (actualización) de las figuras disponibles en la base de datos y un temporizador regulable que cambie las figuras cada cierto tiempo.

Para ello se proponen tres opciones. La primera es tener una base de datos limitada pero actualizable periódicamente mediante acceso por la red a bases publicas de fotografías. El inconveniente de esta opción radica en que la necesidad de que tales fotografías sean de carácter aleatorio, no se puede suplir del todo bien, ya que sería necesario indicar los criterios de búsqueda. Como alternativa, se podría pensar en utilizar palabras que se han introducido en la descripción de las funciones en el momento de definir el modelo funcional, pero esto limitaría un poco el carácter de aleatoriedad necesario para lograr establecer nuevas relaciones.

La segunda opción es utilizar uno de los llamados web-robots (Koster, 1995) que son programas diseñados para automatizar la búsqueda y actualización de información en la

red, previa definición de los criterios. Hay algunas versiones libres en la red, como el Bloodhound y el Collective (Smart Internet Solutions, 2004), DownLoad Express, RoboFox, etc. (Koster, 1995). En este caso, tendría que generarse los criterios de búsqueda previamente y en forma aleatoria, para lo cual se puede utilizar un algoritmo de generación aleatoria de palabras (semejante al de Axon Idea Processor) que luego sirvan para hacer la búsqueda de figuras en el Web y, de esta manera, actualizar la base de datos. Esta opción parece la más viable que la anterior, teniendo en cuenta en primer lugar, que el tiempo de búsqueda y descarga de imágenes es grande y que si se efectúa en tiempo real (durante la ejecución de una sesión práctica) puede cansar al usuario, y en segundo lugar debido a muchos usuarios podrían estar conectados a la red en forma temporal y no permanentemente.

La interfase para esta herramienta tendría la forma mostrada en la Figura 10.11. Contiene el elemento principal que es la gráfica, una ventana para anotar las ideas que puedan surgir (forma escrita) y los botones de opciones, dentro de las cuales debe incluirse una que permita acceder posteriormente a la imagen actual, considerando que se puede dar el caso (como efectivamente sucedió varias veces durante la fase experimental) que una imagen que no genere una idea inmediatamente, puede resultar interesante como para revisarla nuevamente mas adelante.

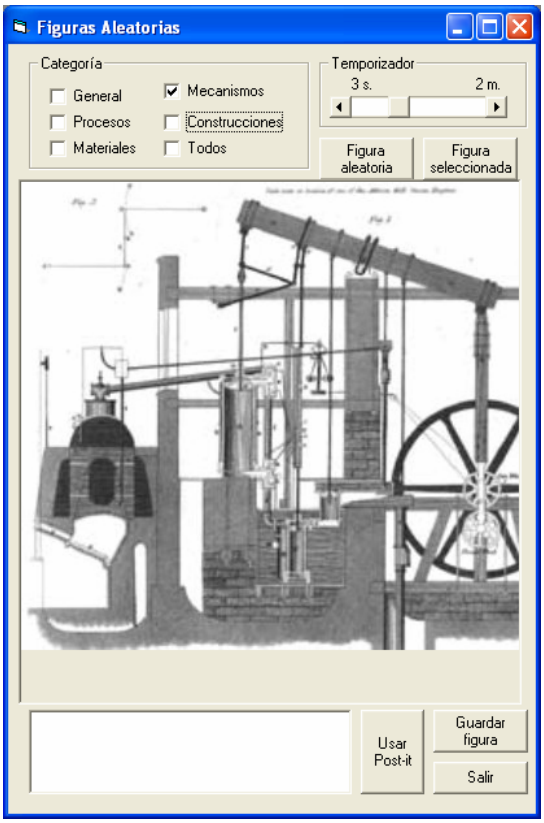

Figura 10.11 Interfase cuando se utiliza el generador de figuras aleatorias.

Dentro de la base de datos de las imágenes se incluirán varias categorías que el usuario seleccionara a conveniencia. La principal será una de tipo libre, es decir imágenes completamente aleatorias tomadas de la vida cotidiana, como la que tiene el programa Brainstorming toolbox. Pero habrá otras especializadas por temas: mecanismos, materiales, construcciones, procesos de manufactura, etc. De esta manera, si el usuario necesita ideas para desarrollar un dispositivo mecánico, por ejemplo, tendrá acceso a ilustraciones que pueden llevar a ideas nuevas.

Dentro de aquellas bases de ilustraciones se puede incluir una de las técnicas de creatividad que aunque no estuvo presente en ninguno de los programas estudiados, puede ser muy útil para dar ideas de solución a problemas técnicos. Se trata de la biónica o estudio de la naturaleza aplicada a la creación de lo artificial. La presentación gráfica de elementos significativos de la naturaleza puede sugerir soluciones importantes para el diseño de nuevos productos, aplicando alguna de las principales características de los sistemas biológicos: su miniaturización, su sensibilidad, su alto grado de flexibilidad, su capacidad para adaptarse a entornos variables y su alto grado de fiabilidad (Saugar, s.f).

10.4.2 Palabras aleatorias

Es la versión textual de la anterior técnica, es decir, se trata de que el programa genere palabras aleatorias e invite al usuario a buscar ideas a partir de ellas. Es muy simple y su presentación en la interfase sería como se muestra en la Figura 10.12 donde aparece la opción de trabajar con diferentes tipos de palabras como sustantivos, verbos, oficios, lugares, etc., en forma individual o combinada.

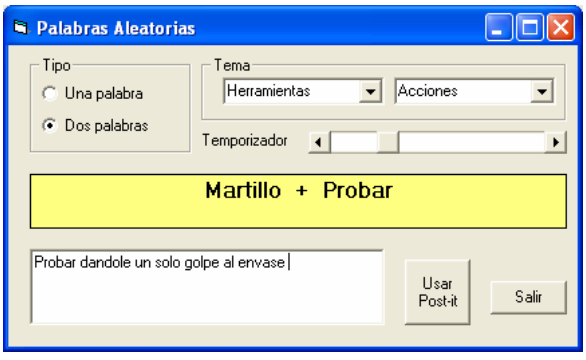

Figura 10.12 Ejemplo de la interfase para aplicar la técnica de palabras aleatorias

10.4.3 SCAMPER

SCAMPER es una versión de lista de chequeo estructurada en seis categorías, cuyas primeras letras dan significado al nombre de la técnica. Así, S significa sustituir, C combinar, A adaptar, M modificar, P utilizar para otras funciones (put on another use, en inglés), E eliminar y R reversar o utilizar inversamente. La lista de chequeo intenta hacer que el usuario aplique alguna de las operaciones señaladas al producto o sistema, utilizando preguntas.

Por ejemplo para la operación Combinar se podrían utilizar preguntas como las que muestra la Figura 10.13. Allí también se muestran las alternativas que puede tener la técnica y la forma de interacción con el usuario.

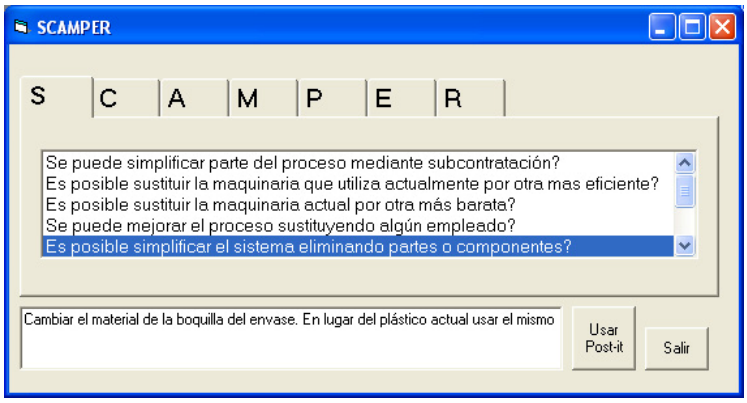

Figura 10.13 Ejemplo de la interfase para aplicar la técnica de SCAMPER

10.4.4 Juegos de rol

Esta técnica es empleada en los programas Braisntormign y ThougthPath y tuvo buenos resultados aunque se señalaron algunas deficiencias. El estudio experimental realizado permite ver que cuando es el programa el que define el personaje se pierde interés por parte del usuario porque muchas veces se trata de personajes que no tienen nada que ver con el problema lo que obliga a hacer suposiciones demasiado ficticias.

El método que aquí se propone para utilizar el juego de rol implica que el usuario seleccione el personaje, pero siguiendo cierto orden y pasando por ciertos niveles, tal como lo ilustra la Figura 10.14. Usuario directo es el cliente final del producto, no es el que lo compra sino el que lo usa, mientras que el usuario indirecto es aquel que sin utilizar el producto se beneficia (o se perjudica) de alguna manera. El fabricante o productor es tanto el empresario como el obrero que usa la máquina para realizar el

objeto. Y el científico es el que conoce la ciencia y la técnica aplicada al objeto. El objetivo del juego de rol es que el usuario del software piense como lo haría cada uno de los personajes identificados sobre el problema que se quiere resolver, en especial tratando de responder a la pregunta «¿qué características o cómo le gustaría que fuese el producto ideal?».

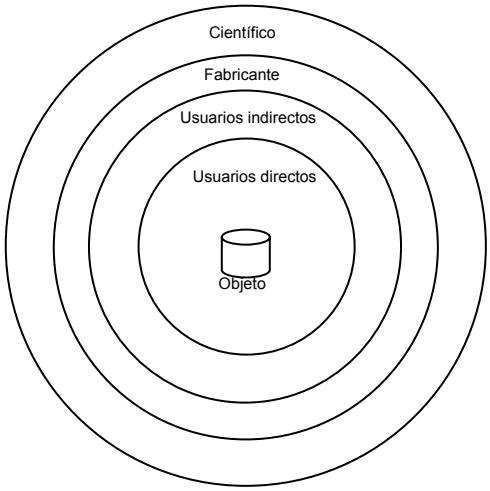

Figura 10.14 Tipos y niveles de personajes que intervienen en el sistema

El software tendrá definidos los tipos de personajes de manera que se puedan seleccionar los pertinentes para cada nivel. Una vez seleccionados el programa los presentará en forma secuencial de manera que se cubra todo el espectro de usuarios del producto. Un ejemplo de la apariencia de la interfase se muestra en la Figura 10.15.

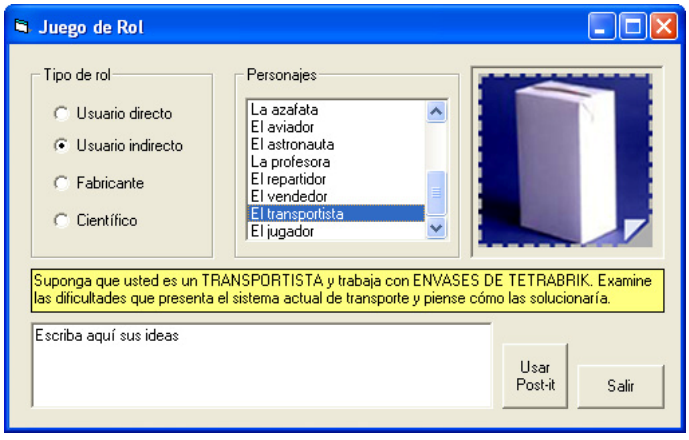

Figura 10.15 Ejemplo de la interfase aplicando la técnica de juegos de rol

10.4.5 Preguntas

La Figura 10.16 muestra la ventana de aplicación de esta técnica. Se propone una serie de Temas para que el usuario seleccione los que interesen, y un cuadro de Funciones mostrando las funciones asociadas al problema que se esta resolviendo en ese momento. Al hacer la selección, el programa combinará la base de datos de preguntas asociadas a la temática con la función respectiva, generando la lista de preguntas que se muestran en la ventana inferior.

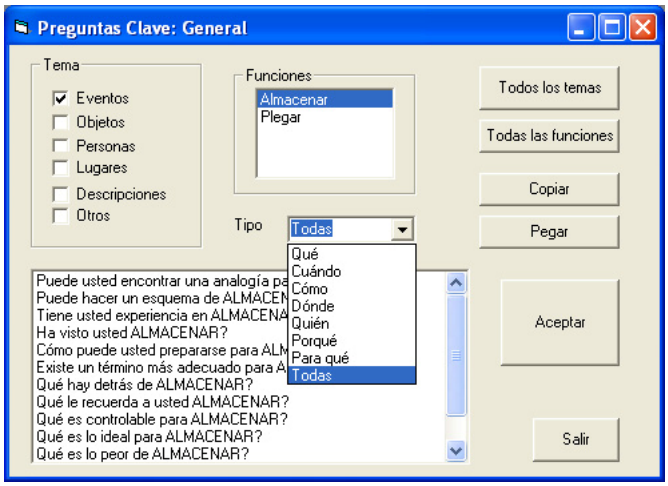

Figura 10.16 Ejemplo de la interfase aplicando la técnica de pregunta

10.4.6 Principios inventivos y contradicciones

Una de las herramientas mas importantes de la metodología TRIZ es la llamada «matriz de contradicciones» propuesta para resolver las incompatibilidades que se suelen encontrar durante la búsqueda de soluciones técnicas. Aunque en las sesiones experimentales no resultó muy útil, se entiende que la razón ha sido la falta de entrenamiento suficiente en el uso de la metodología, pero la técnica en sí misma es valiosa y por ello se propone que haga parte del software.

Cuando se busca soluciones novedosas a un problema técnico es frecuente encontrarse con dos tipos de situaciones contradictorias, que el TRIZ las denomina contradicciones físicas y contradicciones técnicas.

a. Contradicciones físicas

Las contradicciones físicas se identifican cuando para solucionar un problema se requiere que una misma característica del sistema esté presente con dos valores o en dos estados

opuestos. Puede ser, por ejemplo, que se requiera ver con claridad un elemento del sistema, pero también se necesite que permanezca oculto; o que sea imprescindible que un tubo tenga un diámetro grande pero que a la vez sea pequeño. Para este tipo de contradicciones, la metodología propone aplicar alguno de los siguientes cuatro principios denominados «de separación»: basada en condiciones, en el tiempo, en el espacio y entre las partes y el todo.

En el primer caso ocurre cuando se logra que la característica en cuestión este presente bajo cierta condición particular, pero ausente bajo otra condición. En el caso del tubo, podría pensarse que sea grande cuando tenga cierta temperatura, pero que sea pequeño cuando la temperatura sea menor. El segundo principio de separación, en el tiempo, se aplica cuando la característica debe estar presente en un momento determinado, pero ausente en otro momento. Un ejemplo típico es el tren de aterrizaje de los aviones, presente cuando está en tierra y ausente cuando va volando. El tercer principio es el de la separación en el espacio, es decir, que la característica este presente en un lugar determinado pero ausente en otro lugar. Por ejemplo, que el tubo aquel tenga un diámetro grande en cierta trayectoria y uno menor en otra. Y el cuarto principio (entre las partes y el todo) se aplica cuando una característica tiene un valor a nivel de sistema completo, pero el valor opuesto o diferente a nivel de componente. Un ejemplo clásico es el del alambre conductor de electricidad, que cuando está completo (con el aislante plástico) no es conductor exteriormente.

El software integrará una herramienta que le facilite al usuario la identificación del principio adecuado para buscar la solución. Se tratará básicamente de una herramienta de tipo informativo con la explicación breve del principio y con ejemplos de aplicación que lo ilustren. Esto se debe a que realmente en la sustentación teórica del principio no existe aun un método sistemático o automatizable que permita la rápida identificación de la contradicción física, quedando supeditada a la práctica y al conocimiento del usuario. Así que, se facilitará este proceso con información y ejemplos. Lo que sí debe incluir el programa es la integración como ejemplos adicionales las soluciones logradas durante las sesiones anteriores, actuando así como memoria empresarial.

b. Contradicciones técnicas

Las contradicciones técnicas se presentan cuando la mejora de una característica de un sistema, implica la degradación de otra característica del mismo sistema. Por ejemplo cuando se aumenta la dimensión de una pieza se logra mejorar su resistencia pero a su vez el peso y el coste también aumentan. Típicamente el ingeniero soluciona estas contradicciones haciendo un balanceo o transacción de los parámetros, es decir, asumiendo como buena una solución intermedia, sacrificando el valor del parámetro

deseado para no incrementar el valor de la otra característica no deseada. En el mejor de los casos puede aplicar algún algoritmo de optimización para obtener la mejor solución bajo las restricciones impuestas.

El TRIZ por su parte propone la solución de conflictos entre parámetros mediante la aplicación de principios inventivos identificados mediante el análisis de las soluciones encontradas básicamente en el estudio de patentes. Para ello define 39 clases de parámetros que pueden entrar en contradicción entre sí y para su solución propone la aplicación de alguno de sus 40 principios inventivos.

El software integrará la herramienta denominada «matriz de contradicciones» en forma segmentada para facilitar la visualización y selección, ya que cuando se presenta la matriz completa (39x39) resultaría extenuante. La Figura 10.17 muestra la interfase de la matriz propuesta, en contraste con la utilizada por el software estudiado. Se puede apreciar que los parámetros han sido agrupados por categorías y que los principios de solución sugeridos se presentan con su explicación descriptiva y con la opción de visualizar ejemplos típicos. Evidentemente la construcción de la herramienta demandará un trabajo importante de búsqueda de ejemplos, teniendo en cuenta que los ejemplos publicados como lo expresa (Nakagawa, 1999) son muy escasos, principalmente debido a que se desarrollan en empresas que no publican sus resultados o si lo hacen, no incluyen los aspectos más relevantes.

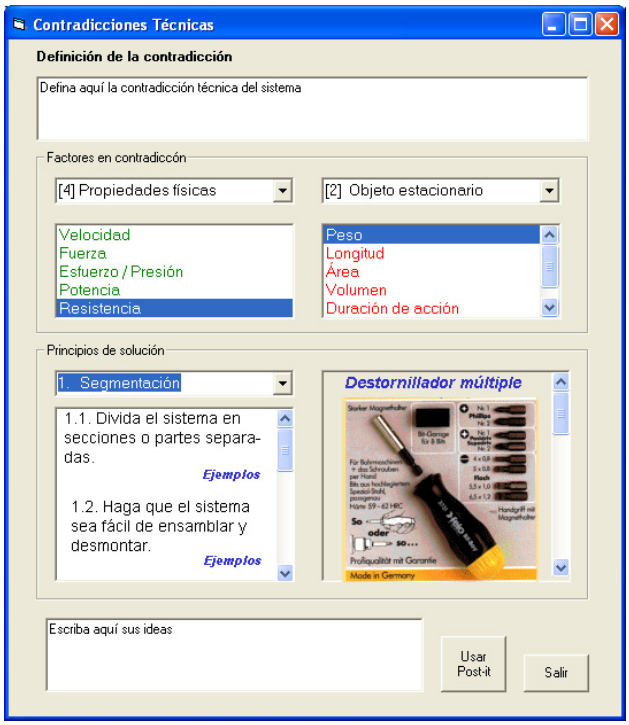

Figura 10.17 Ejemplo de la interfase para la aplicación de la matriz de contradicciones

10.5 Bases de datos

Además de los módulos de estímulo a la creatividad, es muy importante la asistencia al ingeniero en el suministro de información adecuada y oportuna. Por ello, las bases de datos que aquí se proponen resultan altamente relevantes para un software como el que se está proponientoa. Tal como lo señala Ullman (2002) la importancia de que el sistema de soporte a las actividades del ingeniero incluya prioritariamente el manejo de información en lo que él denomina «ambiente externo» al diseñador, radica en la posibilidad de servir de «extensión» de la memoria (de corto y largo plazo) para la actividad propia del diseño, y también como «almacén» de la actividad desarrollada durante el proceso para que pueda ser utilizada posteriormente y no se corra el riesgo de perder lo que aquí se podría denominar la «memoria institucional» de la compañía.

De acuerdo con los principios operativos expuestos las bases de datos que se proponen como parte integrante del software tienen que ver con gestión de la información para: identificación de funciones y atributos de funciones, determinación de los recursos del sistema, revisión de patentes, almacenamiento de ideas propias o «banco de ideas» y estado de desarrollo técnico de la competencia o benchmarking. Las características de cada una de ellas se exponen a continuación.

10.5.1 Recursos

Aunque los principios de estructura y de utilización de esta base ya han sido expuestos anteriormente, cuando se señalaba su importancia en la identificación de los recursos del sistema, es conveniente precisar aquí algunos detalles.

La base de datos que contiene los recursos puede estar organizada inicialmente por las siguientes categorías:

- Sustancias y materiales
- Energías
- Funciones
- Reservas

Para las dos primeras categorías mencionadas la información sobre los recursos se dispondrá en tres niveles diferentes: identificación, propiedades y transformación o modificación. De esta manera, es posible explorar en profundidad las opciones que los recursos disponibles en el sistema ofrezcan para la solución del problema de diseño. La

categoría de funciones y de reservas tienen una estructura diferente, semejante a una lista de chequeo, aunque con algunas variantes.

a. Sustancias y materiales

En esta categoría se incluyen sustancias y materiales líquidos, sólidos y gaseosos. Cada una de estas clases de materiales a su vez se subdividen siguiendo formas estandarizadas de clasificación. Por ejemplo, los materiales sólidos se clasifican en cinco grupos: metales, cerámicos, polímeros, semiconductores y materiales compuestos. Los materiales de cada uno de estos grupos poseen estructuras y propiedades distintas, por lo cual la base de datos dará una información esencial sobre tales características. Además en este nivel se mostrará un listado de los materiales que componen cada uno de los subgrupos. Ejemplos del primer nivel se muestran en la Tabla 10.1.

Tabla 10.1 Ejemplos de sustancias

Gases	*Líquidos*	*Sólidos*
• Aire atmosférico • Gases inertes • Gases inflamables • Gases corrosivos • Gases criogénicos • Gases oxidantes • Gases tóxicos	• Agua • Soluciones diversas en agua • Combustibles • Aceites • Ácidos • Lodos • Pinturas • Tintas • Etc.	• Metales • Cerámicos • Polímeros • Semiconductores • Compuestos.

En el segundo nivel se procede a hacer una descripción de las propiedades de cada sustancia o material. Por ejemplo, para el caso del aire atmosférico se suministraría la siguiente información:

- *Composición:*
  Nitrógeno: 78%
  Oxigeno: 21%
  Argón: 0.93%
  Otros: trazas.
- *Propiedades:*
  Densidad (kg/m3): 1.2
  Presión (Pa): 101,325
  Conductividad térmica: 9.807

Y en el tercer nivel se presentan las transformaciones o modificaciones posibles de la sustancia. Por ejemplo, para el agua: Evaporación, solidificación, condensación, dilución, Ionización, etc.

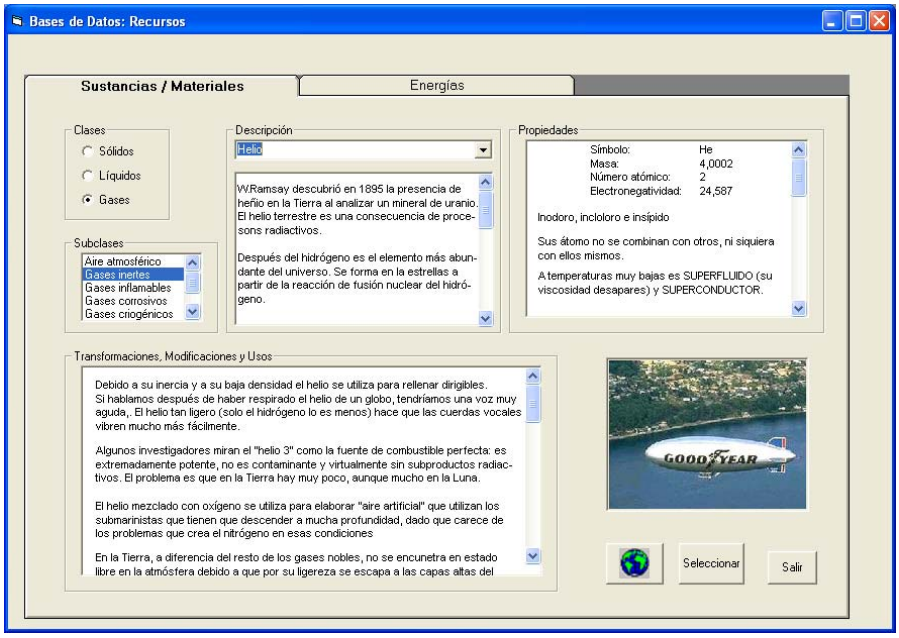

Figura 10.18 Ejemplo de la interfase para la base de datos de Recursos y Energías

Cada uno de los registros de la base de datos estará acompañado de explicaciones concretas de manera que el usuario pueda acceder fácilmente a la información relevante en cualquier momento.

b. Energías

Como en el caso anterior, en el primer nivel se hará la identificación de la energía, por ejemplo: eléctrica, solar, térmica, hidráulica, muscular, nuclear, etc. En el segundo nivel se describen sus propiedades más importantes, por ejemplo para el caso de la energía muscular se dispondría de datos sobre la fuerza máxima y la frecuencia que puede hacer una persona normal. En el tercer nivel se mostrarían los procesos involucrados en la producción y la gestión de energías, como por ejemplo, los procesos de transferencia de calor y formas de controlarlos.

c. Funciones y atributos de funciones

Esta es una base de datos cuyo objetivo es facilitar la identificación de funciones mediante verbos activos y de sus atributos mediante adverbios. Por lo tanto, se trata de una base de datos simple de un solo nivel de estructura. La interfase permitirá seleccionar cuántas funciones y modificadores sean necesarios almacenando la posición o índice del registro respectivo, de manera que sea factible su recuperación en cualquier momento. La Figura 10.19 muestra la forma que tendría la interfase.

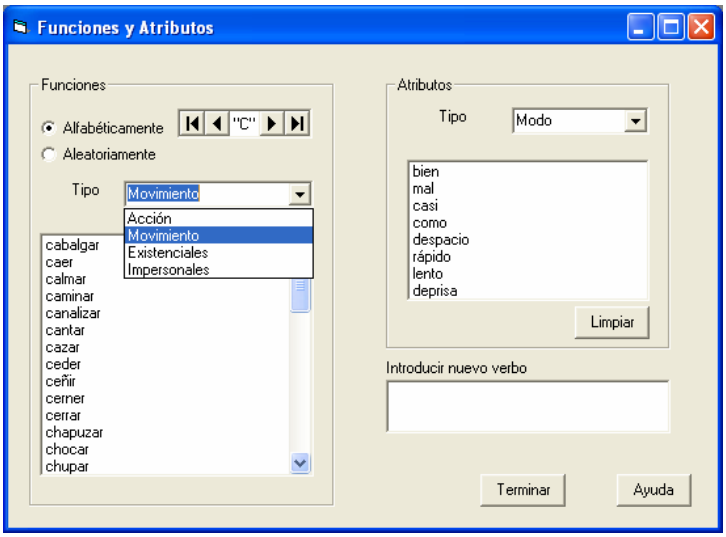

Figura 10.19 Ejemplo de la interfase para la base de datos de funciones y atributos

En caso de que el usuario directamente ingrese el verbo a través de la interfase gráfica de modelación funcional, el motor de la base de datos se encargará de hacer la búsqueda del índice del registro coincidente si existe y si no, lo incorporará como nuevo registro.

El acceso a esta base se podrá hacer a través del menú de recursos que aquí se está presentando, pero también estará disponible directamente desde el menú general, debido a que deberá estar disponible durante todo el proceso, desde la primera fase cuando se realiza la modelación funcional hasta la última cuando se pasa a la evolución funcional controlada.

Una forma adicional de conseguir información relacionada con este tema es acceder por la red a bases de datos de funciones, semejantes a la propuesta por Creax (2000), la cual aunque está en fase de desarrollo resulta de utilidad a la hora de encontrar funciones técnicas especiales. En la maqueta del software que se presenta en el CD4 anexo, se puede ver la funcionalidad de esta herramienta y aunque este en ingles resulta muy intuitiva y sencilla de trabajar.

d. Reservas

Aquí se hace referencia a aquellos recursos materiales o no materiales que pude tener un sistema, como por ejemplo: tiempo libre, espacio no ocupado, iluminación disponible, cambios naturales de temperatura, aislamientos (térmico/acústico) del recinto, información, conocimiento, experiencia, equipos con tiempos muertos, etc. La base, para

esta categoría, actuaría como una lista de chequeo para que el usuario verifique la existencia y pueda tenerlos en cuenta en su búsqueda de una solución. La Figura 10.5 presentada anteriormente muestra esta lista de chequeo (p.276).

10.5.2 Patentes

Las patentes se constituyen en una fuente primaria de información para la innovación tecnológica, debido a que son la forma de presentar en público las ideas inventivas y permiten hacer el «rastreo» del estado de la técnica a nivel mundial mas actualizado posible. Se puede afirmar que las patentes se constituyen en una fuente invaluable de información técnica porque:

- Es actualizada. Debido a que las empresas desean proteger sus inventos, proceden a solicitar la patente lo más rápidamente posible. A partir de allí a mas tardar en 18 meses estas solicitudes se publican y por lo tanto representan la primera información pública disponible. Es decir, son la fuente más actualizada de información tecnológica.

- Es detallada. Los documentos de patentes deben describir en forma detallada y completa la invención, por lo cual generalmente son redactas por personas experimentadas en la materia. Este requisito explica el hecho de que el 70% de la información de patentes no está disponible en ninguna otra fuente.

- Es accesible. Los documentos de patentes tienen un formato unificado y un sistema de clasificación internacional, que permite una rápida comprensión y accesibilidad de la información que contenga.

- Está disponible. Existen bases de datos y colecciones completas de patentes disponibles para que cualquier persona interesada puede recurrir a ellas para encontrar en poco tiempo una gran cantidad de información tecnológica relevante. Además muchas de estas oficinas ofrecen la posibilidad de acceder por Internet a sus bases de datos facilitando así la consulta remota de los documentos de patentes.

- Es exhaustiva. Se estima que más del 80% de todos los conocimientos técnicos en el mundo se pueden encontrar en las patentes (Segura, 1998).

Por todas estas características, un software que pretenda asistir al ingeniero en la fase de conceptualización de productos debe incluir algún tipo de herramienta de búsqueda de patentes.

La opción mas sencilla consiste en incluir en el software botones de acceso a las principales bases de datos de patentes a nivel mundial, como son: esp@cenet a nivel europeo (European Patent Organisation, 2001), USPTO de los Estados Unidos (United States Patent and Trademark Office, 2001), la CIPO de Canadá (Canadian Intellectual Property Office, 2001), la japonesa JPO (Japan Patent Office, 2004), entre otras. Esta opción es la implementada en el software Creax Innovation Suite, que fue el único programa de los estudiados que incluye el tema de patentes. Evidentemente, el usuario debería tener conocimientos y experiencia previa en la búsqueda y revisión de patentes, debido a la cantidad de información disponible en cualquiera de las bases de datos de patentes señaladas, que puede resultar abrumadora para el usuario.

El software TechOptimizer (Invention Machina, 2002), que también se basa en el TRIZ, implementa como alternativa una herramienta de búsqueda semántica automática denominada «Goldfire Innovator™» desarrollada con tecnología semántica indexada que se basa en la matemática lingüística (Verbitsky, 2004). Se fundamenta en la realización de un análisis del lenguaje natural en cuatro niveles diferentes: reconocimiento de la palabra, análisis del léxico, análisis sintáctico y análisis semántico. El primer nivel ya está desarrollado ampliamente y el ordenador lo ejecuta sin problema, por la digitalización de la palabra ya sea hablada o escrita. En el segundo nivel se hace una división de la oración en palabras individuales para identificar el tipo de cada una mediante una comparación con un diccionario interno. El análisis sintáctico estudia la gramática de la frase para identificar su estructura sintáctica y precisar las palabras. Finalmente, el análisis semántico identifica el significado del texto en términos de elementos semánticos como sujeto, verbo, sustantivo, adjetivo y adeverbio.

Como resultado de este análisis el ordenador puede identificar la estructura y el contenido de las frases que el usuario ingresa y efectuar a continuación una comparación con las bases de datos externas (patentes) o internas para buscar coincidencias y entregar resultados filtrados. Por ejemplo, si el usuario ha definido su problema con la pregunta «cómo puedo comprimir la caja de cartón?», el ordenador identifica la palabra «comprimir» como el verbo (la acción), el sustantivo (objeto) será la «caja de cartón» y el sujeto será el elemento a buscar. Este mismo análisis lo realizará buscando coincidencias de la acción y el objeto en la base de datos y recuperará, por lo tanto, solamente la información relevante. El «Goldfire Innovator™» contiene ya tal base de datos construida a partir de la aplicación del análisis a «toda la colección de patentes a nivel mundial» (Verbitsky, 2004, p.6).

Evidentemente que aquella herramienta no será posible utilizarla en el software actual. Pero la idea es útil para buscar alternativas. Una de ellas es enlazar con software especializado para la búsqueda semántica de información, como por ejemplo:

- MicroPatent®. Que proporciona acceso on-line a servicios de gestión de información desde búsqueda de patentes, análisis masivo de texto y trazado de árboles genealógicos de patentes (Aureka, 2004).
- Matheo Patent . Un software diseñado para explorar rápida y profesionalmente la base de datos de patentes europeas (esp@cenet). Colecciona automáticamente las patentes de acuerdo con los criterios definidos por el usuario, construyendo y manteniendo actualizada una base de datos propia.

Y muchas otras opciones disponibles como PatStat de Derwent, Tetralogie de Atlas, etc., utilizadas también para realizar vigilancia tecnológica y minería de datos.

Una tercera alternativa sería la adecuación y utilización de un web-robot que automáticamente navegue por la red buscando y recuperando documentos previamente referenciados, tal como se explicó anteriormente (Koster, 1995). Este es un tema que sería necesario explorar con profundidad.

Las dos últimas alternativas señaladas se salen del alcance de esta investigación y aunque no se descarte su potencial utilización en el software, se prefiere proponer una alternativa intermedia que utilice las bases de datos públicas de patentes como lo hace el Creax, es decir mediante un botón de acceso directo.

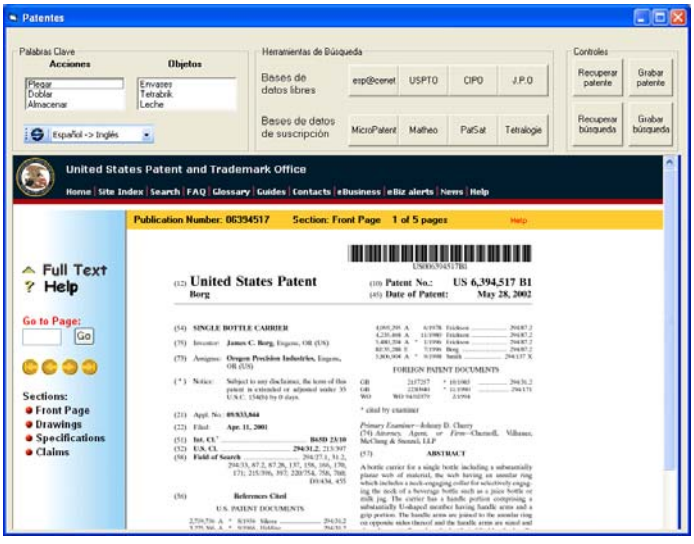

Figura 10.20 Ejemplo de la interfase de búsqueda de patentes

Sin embargo se aprovechará el esquema de definición funcional que se ha propuesto como modo general de trabajo en el software para identificar las funciones (verbos) y los objetos (sustantivos) para ser utilizados como criterios de búsqueda.

Por ello, el programa ofrecerá al usuario una ventana donde se encuentre la descripción funcional del problema que se esté resolviendo y se invitará a utilizar las palabras clave para la búsqueda y refinación de las patentes que se vayan encontrando. Una muestra de los que sería tal interfase se aprecia en la Figura 10.20.

10.5.3 Benchmarking secundario

Aquí se hace referencia a la conveniencia de disponer de una base de datos para la gestión de la información de la competencia. Es lo que se denomina «benchmarking secundario» (Instituto Español de Comercio Exterior, 2003), que consiste en la recopilación de información de dominio público de un sector de actividad, empresas competidoras, mercados, clientes, proveedores, etc., con el propósito de descubrir y mantenerse actualizado en las tendencias que aquellas muestran. Esto es parte del proceso denominado vigilancia tecnológica, aunque no se pretende que este software desarrolle tal actividad en forma tan sofisticada como lo hacen los especialistas.

Figura 10.21 Ejemplo de la interfase para búsqueda de catálogos de productos

El objetivo es aprovechar la creciente disponibilidad en la red de catálogos electrónicos para recuperar y mantener actualizada la información sobre las tendencias de productos

similares a los de la propia empresa. Al igual que en el caso de las patentes podría eventualmente utilizarse alguna herramienta que automatice el proceso de búsqueda y actualización de información relevante. Para esta labor sí parece ser más eficiente el uso de los web-robots (Koster, 1997).

En forma alternativa, se puede sencillamente diseñar la interfase adecuada para realizar meta-búsquedas manualmente, como lo muestra la Figura 10.21.

10.5.4 Banco de ideas propias

Puede suceder que ideas que se generen durante la ejecución del software para un problema en concreto no sean seleccionadas o consideradas útiles en ese momento y sin embargo ser muy buenas soluciones para otros problemas que puedan abordarse posteriormente. Por ello resultará útil conservar tales ideas almacenadas y disponibles en el software.

La base sencillamente almacenará los datos relevantes de la sesión en curso: fecha, usuario, nombre del problema e ideas generadas. Estas últimas representadas por el modelo funcional evolucionado con todas las estructuras de solución que se propusieron. De esta manera, se podrá indexar y recuperar la información por cualquiera de los cuatro criterios. La forma de la interfase para esta base de datos se enseña en la Figura 10.22.

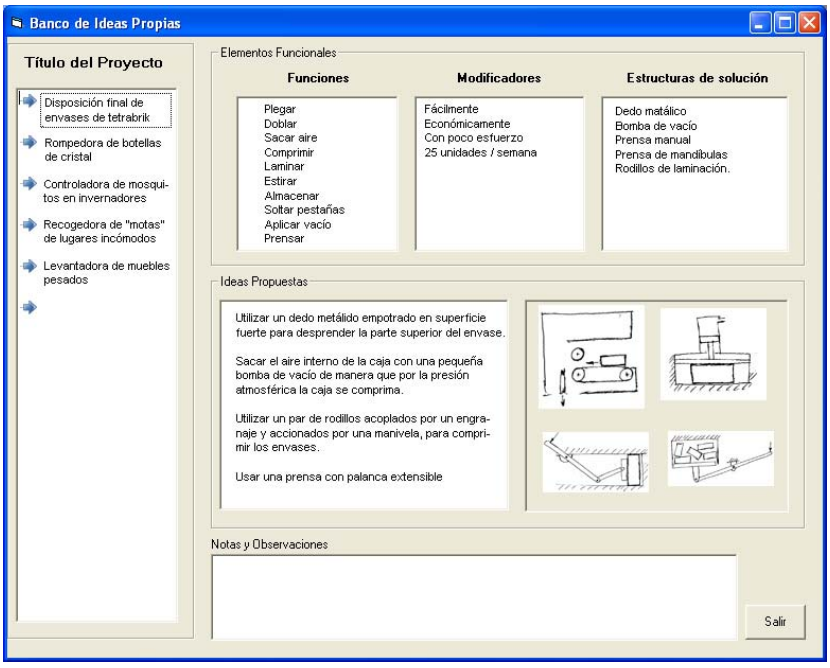

Figura 10.22 Ejemplo de la interfase del «banco de ideas»

10.6 Elementos de usabilidad

Durante toda la estructuración del software se ha buscado un comportamiento consistente con principios de usabilidad reportados en la literatura: pensar en el usuario, crear enlaces entre la aplicación y el mundo real, hacer la aplicación consistente y familiar, orientar adecuadamente el proceso pero permitiendo que sea el usuario el que este bajo el control de la secuencia, etc. (Crow y Cansen, 1998, Dumas y Redish, 1999, Benson y et.al., 2002, Torrealba, 2004). Ahora se completará esa consistencia tratando ya no con el comportamiento sino con la apariencia, esto es, con la forma externa o interfase del software.

Para ello, la sección está organizada en dos categorías principales. La primera aborda las características de tipo globales del software, es decir, aquellas que se pueden identificar a partir de las conclusiones generales del experimento, mientras que la segunda se centra en la aplicación de aquellas características en la forma particular de la interfase propuesta para el software. Es claro que las solas explicaciones verbales son insuficientes para ilustrar las características deseables de la interfase, por lo cual se presenta en el Anexo 5, la maqueta completa del programa que cosiste en una presentación simulada y auto-explicada del funcionamiento del programa.

10.6.1 Elementos generales

Resulta conveniente adoptar un modelo de evaluación ex - ante de la calidad del software aunque se este abordando solamente su forma estructural. Existen varias propuestas en la literatura, pero aquí se tomaran algunos elementos del modelo de McCall por ser uno de los más conocidos y porque ha servido de base para varias propuestas derivadas (Cervera, Nuñez y Bernardo, s.f.).

Tal modelo se sustenta en la hipótesis de que el usuario puede valorar la calidad de un software teniendo en cuenta once criterios diferentes, agrupados en tres ejes principales, tal como se muestra en la Tabla 10.2

Se define la calidad del software como «el grado con el que un sistema componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario» (IEEE, 1990). Esta fase la discusión se centrará en aquellos aspectos relacionados con la visión del usuario, más que en la definición de la estructura y la definición de contenido, que ya fueron expuestas en las secciones precedentes, o en especificaciones técnicas del software, que sería objeto del desarrollo propio del software, actividad que se sugiere realizar posteriormente a esta investigación. Esto

significa que la definición de elementos estructurales que a continuación se tratará, estará centrado en el primer eje descrito en la tabla 10.2: «facilidad de operación», y específicamente en el primer factor, esto es, «facilidad de uso». Además se tendrá en cuenta otro criterio no mencionado explícitamente por el modelo McCall, como es la «utilidad» del software, aquí entendida como el eficiente uso de las técnicas de creatividad y las bases de datos para la generación de ideas.

Tabla 10.2 Modelo de valoración de calidad de software de McCall

Ejes de evaluación	***Factores***	***Criterios***
Operación	Facilidad de uso	Facilidad de aprendizaje Facilidad de operación. Facilidad de comunicación Formación
	Integridad	Control de accesos Facilidad de auditoria Seguridad
	Corrección	Completitud Consistencia Rastreabilidad.
	Fiabilidad	Precisión Consistencia Tolerancia a fallos Modularidad Simplicidad Exactitud
	Eficiencia	En ejecución En almacenamiento
Capacidad de soportar cambios	Facilidad de mantenimiento	Modularidad Simplicidad Consistencia Concisión Auto descripción
	Facilidad de prueba	Modularidad Simplicidad Auto descripción Instrumentación
	Flexibilidad	Auto descripción Capacidad de expansión Generalidad Modularidad
Adaptabilidad a nuevos entornos	Reusabilidad	Auto descripción Generalidad Modularidad Independencia del sistema operativo Independencia del hardware
	Interoperatividad	Modularidad Compatibilidad de datos Estandarización de datos
	Portabilidad	Auto descripción Modularidad Independencia del sistema operativo Independencia del hardware.

a. Facilidad de aprendizaje

Una de las conclusiones relevantes del estudio experimental es la necesidad de disponer de un programa fácil de aprender, es decir, un programa cuya estructura se convierta rápidamente en familiar para el usuario de manera que no demande mucho tiempo de aprendizaje, lo que técnicamente podría denominarse como un programa con una curva de aprendizaje semejante a la mostrada en la Figura 10.23b, es decir, que en corto tiempo se logre un alto dominio del programa.

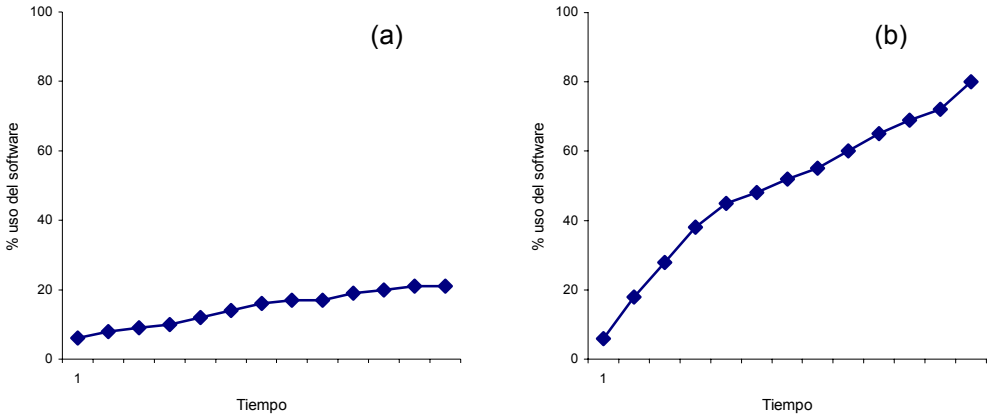

Figura 10.23 Curva de aprendizaje de software típica (a) y deseable (b)

Esto significa que se debe tener en cuenta las actitudes, preferencias y forma de trabajo del usuario. En otras palabras, el diseño del software debe estar centrado en el usuario, tener una interfase amigable que estimule a los nuevos usuarios a adoptar el programa rápidamente como parte de su forma de trabajo y unas herramientas de visualización de opciones intuitivas que permitan adquirir un conocimiento máximo del proceso.

Dumas y Redish (1999) indican que el usuario considera un producto como fácil de aprender, en términos del tiempo que toma hacer lo que él desea, del número de pasos requeridos y del éxito que tiene en predecir la acción correcta a efectuar.

Un claro ejemplo de un software que no reúne estas características, evidenciado por los resultados encontrados en la fase experimental, es el programa que teóricamente era el más adecuado para apoyar al ingeniero de diseño dada su estructura y su metodología específicamente orientadas a este propósito (Creax Innovation suite), y que no mostró el mejor desempeño al compararlo con programas menos sofisticados y de carácter mas genérico. Hay varias razones para estos resultados. Una es la complejidad de la metodología TRIZ frente a otras más intuitivas o más conocidas como el “brainstorming” o los mapas mentales, que obliga de por sí a emplear mayor tiempo de aprendizaje. Pero,

indudablemente la propia estructura y presentación del programa no facilita el proceso, al utilizar iconografía no estandarizada, procesos dispendiosos, interfases cargadas, ayudas poco comprensibles, terminología de la pagina ayuda diferente a la del programa, etc.

El cómo hacer un programa que sea fácil de aprender sigue siendo una pregunta difícil de responder incluso por empresas multinacionales dedicadas exclusivamente al desarrollo de software. Sin embargo, se puede identificar algunas pautas que evidentemente contribuirán a ese logro:

- *Uso de iconografía estandarizada.* Esta es una de las reglas básicas de usabilidad.

- *Utilización del idioma natural del usuario*. Muchas de las técnicas de creatividad hacen un uso intensivo de elementos del lenguaje que pierden sentido cuando un usuario que maneja otro idioma efectúa la traducción. Por ejemplo, el uso de metáforas resulta poco útil por la dificultad que tiene un ingeniero, acostumbrado a leer literatura técnica, para traducirlas.

- El programa Brainstoriming toolbox, por ejemplo, tiene dos técnicas que utilizan este tipo de elementos (false rules y random word) y éstas fueron las menos utilizadas en todas las repeticiones del experimento (4.9% y 12.1%). En los otros programas también se evidenció problemas con el manejo del idioma, en especial cuando se hacía necesario rellenar formularios y aparecían combinadas frases preestablecidas en inglés con palabras en castellano, como por ejemplo en AXON donde se pueden aparecer cosas como “Can *caja de cartón* be rated or ranked?”.

- Por lo tanto, para este tipo de aplicación un software debe ser realizado en el idioma del usuario.

- *Tutorial y ayuda adecuados.* El tutorial como elemento fundamental en el proceso de aprendizaje del uso de software debe mantener un equilibrio adecuado, es decir, ser lo suficientemente completo para cubrir los aspectos fundamentales del manejo del programa, pero lo suficientemente ágil para no fatigar con explicaciones innecesarias. Debe tener ejemplos cercanos al tipo de usuario típico del programa (ingenieros, en este caso), que no sean simples ni tampoco demasiado complejos.

- Este también es un campo deficitario en la mayoría de los programas evaluados (lo cual es cierto para la generalidad de programas de ordenador). Por ejemplo, el tutorial de Axon es muy denso y detallado lo que hace que el usuario se canse pronto al leerlo. El Creax tiene el tutorial on-line, diseñado con muchas animaciones que hace lento su estudio. Además utiliza términos que en el programa no se encuentran o son

diferentes (falta actualización). El Brainstorming llama tutorial al enlace con la página Web del productor, de manera que cuando se quiere acceder a algún tópico se entra a la página con todos los distractores que esto conlleva.

- *Modularidad.* Esto es, la característica de un software que proporciona una estructura de módulos independientes pero «inter-conectables», con lo cual el usuario puede seleccionar los módulos más adecuados para su estilo de trabajo sin necesidad de aprender a utilizarlos todos. En el caso particular de un software de creatividad deseable, los módulos podrían estar estructurados alrededor de las diferentes técnicas de creatividad que pudiese disponer el usuario a voluntad de un abanico de posibilidades bien explicadas. De esta manera se evita la saturación de la interfase con opciones que no interesan al usuario y que en todo caso le trasmiten un mensaje de que el trabajo queda incompleto si no los ha utilizado.

En orden creciente de grado de dificultad de aprendizaje, los cuatro programas evaluados se pueden clasificar como se muestra en la Tabla 10.3.

b. Facilidad de operación

Aunque la presión social por productos software de calidad está aun lejos de la que se percibe en otras clases de productos, es claro que poco a poco se va identificando a un usuario más exigente de programas que sean rápida y efectivamente utilizables. Estos objetivos están muy relacionados con la sencillez y, principalmente, con la facilidad de operación. La tolerancia del usuario al grado de dificultad de operación de un software es muy limitada y esto incide directamente en su aceptación o «fidelización». Aunque un programa sea muy eficaz podría ser rechazado si su uso resultara complicado para un usuario típico.

Es necesario partir de la premisa de que el usuario espera ser más productivo al utilizar el software, lo cual siempre tendrá como trasfondo dos factores relevantes: el esfuerzo requerido y el éxito logrado.

El éxito logrado dependerá de la estructura del software, tema que ya se trató en este mismo capítulo. En cuanto al esfuerzo, el indicador más importante será el tiempo necesario para ejecutar una sesión. Evidentemente, la percepción inicial del tiempo necesario para desarrollarla será determinante en el grado de aceptación del software, y es allí donde el tiempo dedicado a las acciones preliminares cobra relevancia. En el experimento se pudo ver que el tiempo dedicado a estas acciones fue mucho mayor cuando se utilizó software que cuando se trabajo sin él. Evidentemente la exigencia de información preliminar por parte de algunos de los programas evaluados resulto tan

exhaustiva que causó sobre el usuario un sentido de perdida de tiempo importante. Esto fue cierto específicamente para el caso del CREAX, tal como se señala en la Tabla 10.4. Para evitar este tipo de situación se recomienda que el software no solicite información irrelevante para el usuario en esta fase del diseño, donde normalmente se quiere abordar el problema sin entrar en detalles innecesarios. En otras palabras el programa debe facilitar el inicio de la sesión de diseño conceptual tan pronto se abre el programa, como lo hace el Axon Idea Processor.

Tabla 10.3 Grado de dificultad de aprendizaje del software evaluado

Programa	***Características***
Brainstorming Toolbox	Utiliza la técnica de creatividad más conocida.
	Contiene un módulo de estimulo gráfico (random picture) que fue el mas utilizado y el mas eficiente para producir ideas.
	Su estructura permite navegar a voluntad, sin tener que seguir una secuencia estricta por los diferentes módulos.
	La introducción de ideas es simple introducción de texto. No se procesa.
ThoughtPath	Es un programa estructurado para seguir una secuencia más o menos predefinida.
	Usa la técnica de la sinéctica. Menos conocida, pero fácil de entender y aplicar.
	Su estructura es poco flexible. Es necesario seguir una secuencia.
	La introducción de ideas es simple, pero la continua solicitud de introducción de datos o frases causa cansancio del usuario.
	La aplicación de muchos de los estímulos (triggers) se hace difícil al emplearse muchas frases hechas, difíciles de traducir.
Axon Idea Processor	Se basa en la técnica de los mapas mentales, también muy conocida.
	Los elementos de estimulo se encuentran un tanto “ocultos” entre todas las opciones de edición del mapa mental. Por ello cuesta encontrarlos.
	Cuando se inicia un proyecto se puede presentar el síndrome de la “página en blanco”, ya que no existe ninguna fase de introducción.
	No se estructura por módulos.
	Existe libertad y muchas opciones para navegar y construir los mapas mentales.
CREAX Innovation Suite	Utiliza la metodología TRIZ, compleja de aprender.
	Maneja módulos pero no son independientes. Se debe seguir toda la secuencia propuesta en el programa.
	Algunas herramientas son difíciles de entender.
	Otras herramientas de edición usan iconos diferentes a los convencionales, y esto causa confusión.
	En general los ejemplos que enseña son muy simples.

Tabla 10.4 Información preliminar solicitada por cada programa

Programa	***Información preliminar***
CREAX Innovation Suite	Descripción del problema Pregunta del problema Datos del proyecto: nombre del proyecto, equipo de trabajo, financiador del proyecto, cliente del proyecto. Objetivos: bajo la visión del cliente, del financiador y del equipo. Metas e indicadores: bajo las tres visiones señaladas. Gráfica que ilustre el problema.
ThoughtPath	Nombre del problema. Descripción del problema. Antecedentes del problema.
Axon Idea Processor	Nombre del archivo. Características de interfase inicial: Tipo de archivo, color de fondo, número de niveles (ya están definidos valores por defecto).
Brainstorming Toolbox	Nombre del archivo y de la sesión. Definición del problema. Una frase que resuma el problema a resolver.

10.6.2 Características de la interfase

Es importante señalar que la forma adoptada para representar el modelo FBS es heredado del software Axon Idea Processor. En cierta manera el modelo funcional se corresponde con un mapa mental por su forma de estructurar y de interconectar los diferentes elementos.

Esta forma de expresión de ideas disminuye aquella tendencia que hace que un usuario de ordenador fije mucha atención en aspectos formales (estilo de redacción, alineación, tamaño, e incluso ortografía). Cuando se realiza un esquema gráfico en papel un usuario es mucho más tolerante con los errores o imperfecciones del esquema, de lo que es cuando trabaja en el ordenador, porque se ha generado el concepto de que en el ordenador se generan solamente versiones finales de lo que se trabaja y no solo "borradores".

Conviene por lo tanto que el software permita que las ideas no sean expresadas solamente como palabras escritas, sino como elementos con cuerpo y contenido, facilitando una forma rápida y eficaz de expresión de ideas (al menos más rápida que haciendo un esquema o croquis y más eficaz que describirla con solo texto).

En el programa Axon las ideas son objetos que tienen tres características: un identificador, una forma y una acción. Cada una de ellas tiene a su vez muchas opciones de configuración. Por ejemplo para definir la forma de un objeto existen 14 opciones diferentes y para definir la acción ¡63¡ alternativas. Evidentemente tal cantidad de opciones confunde al usuario quien termina por utilizar solamente unas cuantas que le resulten significativas.

Ya se comentó que el modelo FBS consta básicamente de funciones, modificadores funcionales, estructuras de solución y las interrelaciones entre aquellas tres. Por lo tanto, se limitarán las opciones para facilitar la expresión de estos cuatro tipos de elementos, así:

Funciones: Estarán representadas por una elipse con un verbo como identificador y con una ventana de hipertexto que se desplegara a voluntad (al dar doble clic sobre la elipse).

Figura 10.24 Forma de representación de las funciones.

En la ventana se incluirá la definición de palabras clave, que serán utilizadas en el software para búsquedas en la base de datos propia y en la red.

Modificadores funcionales: Representados por una línea sobre la cual va el identificador, que suele ser un adverbio o un adjetivo que cuantifica la función. También admite una ventana de hipertexto para explicar la cuantificación que hace y de ser posible precisar valores.

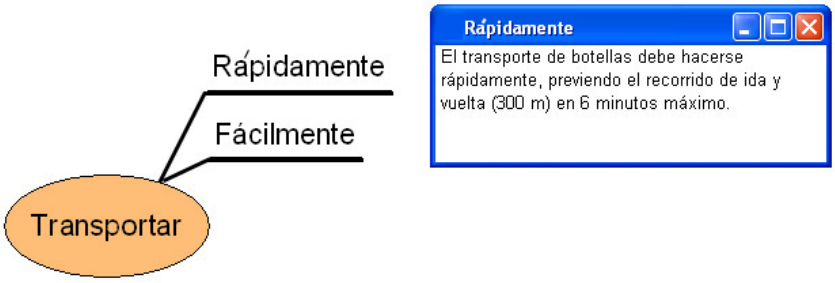

Figura 10.25 Forma de representación de los modificadores funcionales

Estructuras de solución: su forma es rectangular, se identifican con sustantivos y también muestran una ventana de hipertexto-gráfico, con la opción de pasar a la herramienta de esquematización donde se podrá elaborar un gráfica de la solución en forma rápida.

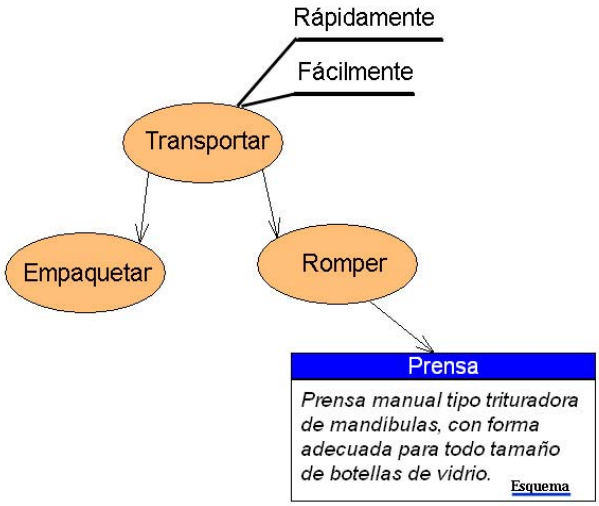

Figura 10.26 Forma de representación de las estructuras de solución

Links: representan las interrelaciones entre los diferentes elementos. Es una flecha para el caso de relaciones funcionales y una línea en el caso de modificadores.

Ya se comentó que cada uno de estos elementos estará acoplado a capas diferentes con el propósito de facilitar la lectura del modelo cuando éste sea muy complejo. Por defecto la visualización será completa, es decir, se verán las tres capas simultáneamente y el usuario a voluntad podrá seleccionar la que desee en cualquier momento. Además la ubicación en la capa respectiva será automática, es decir, cuando el usuario seleccione una función, el programa entenderá que ésta se debe posicionar en la capa de funciones y no en otra. Esto facilitará el trabajo del usuario y evitará que su atención se desvíe del objetivo principal.

Finalmente, la interfase del programa podría tener la forma que se muestra en la Figura 10.27. La descripción detallada de cada componente se presenta en el Anexo 5.

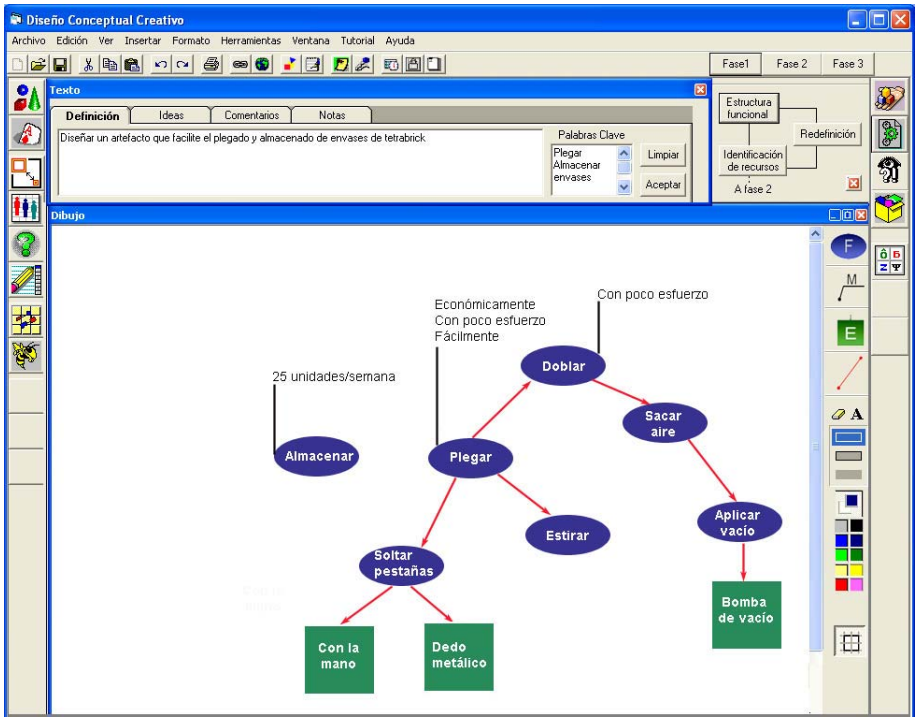

Figura 10.27 Interfase general propuesta para el software

10.7 El problema de la elaboración de dibujos básicos

En la fase de conceptualización es típico que el ingeniero utilice representaciones gráficas que representan formas, que si bien es cierto no están estructuradas completamente, si se constituyen en un medio de expresión muy efectivo. El «lápiz y papel» se ha considerado como herramienta esencial en la creatividad del proceso de diseño (Purcell y Gero, 1998). De hecho la fase experimental de esta investigación coincide con esta afirmación dado que la mayoría de las estructuras de solución tuvieron que ser expresadas a través de esquemas gráficos aunque se hubiesen originado por el estimulo de los programas, porque el medio gráfico permite una rápida interpretación y ejecución de los conceptos que se van generando. Parece existir una interrelación muy estrecha a nivel cerebral entre la mente y la mano. Gross (1988) señala que es a través de los esquemas que el diseñador le da «cuerpo» a la abstracción que caracteriza inicialmente a los conceptos e ideas, mientras que Herbert (1988) identifica el esquema como el medio adecuado para agregar información proveniente de la experiencia previa del diseñador y para recuperar y manipular las representaciones visuales o las imágenes relevantes de un solución potencial.

Es clara, tanto por los reportes de la literatura como por lo evidenciado empíricamente con esta investigación, la importancia de la esquematización en la fase de

conceptualización en el diseño. Tampoco hay discusión en la afirmación de que los programas convencionales de ordenador no han podido brindar una alternativa eficiente para que el ingeniero de diseño pueda ejecutar esquemas básicos conceptuales, caracterizados por su ambigüedad, imprecisión e intención difusa. Los programas CAD, incluyendo los tecnológicamente más desarrollados, tienen un enfoque centrado en el desarrollo de la forma o geometría de las piezas que ya tienen un desarrollo conceptual claro en la mente del diseñador (Ullman, 2002). Así, no existe una aplicación que pueda suplir con eficacia lo que el ingeniero de diseño hace con el lápiz y el papel.

Las alternativas para entrar a dar respuesta a esta insuficiencia en el corto plazo se pueden centrar en el hardware. Evidentemente que la incursión de los llamados «Tablet PC» permite vislumbrar una forma alternativa de interacción usuario-ordenador, que facilitaría la integración de la técnica del «lápiz y papel» empleada por los ingenieros de diseño, ya que el concepto de tinta electrónica permite la interacción más naturalmente.

Existen sin embargo muchas críticas al Tablet PC que lo señalan como pensado más en el usuario ejecutivo, de negocios y que viaja, pero menos para el diseñador y en la necesidad de desarrollar software específico útil para procesar fácilmente los dibujos o esquemas realizados y aunque la publicidad indique su utilidad en estas labores aún no se presentan estudios en la literatura científica sobre su efectividad.

El proyecto denominado «SmartPaper» (Shest y Chen, 2004) que desarrolla actualmente la universidad de Minnesota busca precisamente aprovechar las ventajas de este tipo de ordenadoras para hacer esquemas a mano alzada y luego procesarlos para dejarlos casi como dibujos acabados o al menos disponibles para que con otra aplicación tipo CAD se pueda continuar; pero aún se encuentra en fase de desarrollo.

Como alternativa a la tecnología del Tablet PC pero también relacionado con hardware, es el denominado «bolígrafo digital io2» (Logitech, 2004), utilizado para escribir normalmente y simultáneamente grabar en una pequeña memoria lo escrito para luego descargarlo al ordenador mediante una conexión tipo USB. Tampoco existe información sobre estudios realizados con este dispositivo, pero aparenta ser una solución intermedia entre el Tablet PC y la situación actual.

Una alternativa que puede ser muy interesante, y que se encuentra en estudio actualmente, es la denominada «virtual claying» (Jansson y Vergeest, 2002; Rusák, 2002) consistente en la modelación con arcilla virtual como medio de la expresión conceptual de la forma de elementos. El método permite simular el trabajo de modelación con sólidos, basado en los principios físicos observables de la mecánica de los cuerpos deformables: movimiento de translación y de rotación, colisión y deformación (por

tensión, compresión, flexión, torsión, pandeo y fractura). El proyecto es adelantado por la Universidad de Delft en Holanda y promete ser una alternativa a tener en cuenta en aplicaciones informáticas para el diseño.

Por el momento, la única alternativa viable y que, por lo tanto, deber estar presente en el software que se propone en esta investigación es la utilización de una herramienta de graficación convencional tipo «Paint» de Microsoft, dejando la posibilidad de integrar en el corto plazo alguna herramienta de dibujo de esquemas, más natural como las anteriormente expuestas.

10.8 Conclusiones del capítulo

Se ha presentado en este capítulo la estructura principal y los elementos formales básicos de lo que se puede constituir en una herramienta informática para asistir al ingeniero de diseño en la fase de conceptualización de nuevos productos. Para ello se han utilizado los principales elementos del modelo FBS como instrumento de guía del proceso, integrando con él, los principales argumentos de la metodología TRIZ.

La razón para utilizar estos dos modelos obedece a la conveniencia detectada durante la evaluación experimental, de proveer una estructura que guíe al usuario en forma más o menos sistemática, pero que mantenga a su vez la suficiente flexibilidad como para hacer cómodo el proceso, brindando mayor naturalidad. Por esta última razón no resulta conveniente proponer la estructura totalmente orientada por la metodología TRIZ debido a su dificultad de aprendizaje. De hecho se ha tratado de evitar el manejo de muchos de los términos utilizados por los practicantes de tal metodología, proponiendo en su lugar la práctica de un lenguaje más natural y sencillo, que también representa una propuesta de forma de trabajar más intuitiva.

El proceso propuesto para el desarrollo del software dirige al usuario por tres etapas bien diferenciadas pero perfectamente interconectadas, tal como se explicó en detalle. Cada una de ellas estará asistida por dos tipos de herramientas complementarias como son los módulos de estímulo creativo y las bases de datos.

Las técnicas de creatividad disponibles como módulos de estímulo para la generación de ideas, son: mapas mentales, listas de chequeo, sinéctica, figuras aleatorias y SCAMPER. Se ha tratado de denominarlas en forma adecuada para el lenguaje comúnmente utilizado por el ingeniero, buscando con ello establecer puentes adecuados y romper el

escepticismo que normalmente tiene este tipo de usuarios para utilizar herramientas que no son eminentemente técnicas.

Por otra parte, las bases de datos que se proponen dan lugar a un manejo eficiente de la información que el ingeniero requiere en un proceso de diseño, cubriendo una amplia variedad de opciones y de áreas de conocimiento. La implementación práctica de algunas de ellas (las más complejas) requerirá mayor investigación, si se quiere lograr una automatización eficiente, pero queda claro que son perfectamente viables.

En cuanto al diseño externo o formal del software se ha sugerido mantener los criterios básicos de usabilidad, que son determinantes a la hora de aprender a utilizarlo, de sacarle todo el provecho y de asociarlo al ambiente normal de trabajo. La maqueta descriptiva de las diferentes opciones propuesta se presenta en el CD4 anexo con un diseño auto-explicativo para dar mayor claridad a la propuesta.

Finalmente se abordó el tema la elaboración de descripciones gráficas (bocetos o esquemas), dada su importancia en diseño demostrada en la fase experimental que corrobora las afirmaciones de la literatura especializada. Se dieron algunas pautas de solución que se han propuesto recientemente y algunas que están en fase de desarrollo.

Se puede decir, para concluir, que el experimento junto con los elementos teóricos recopilados previamente y la experiencia profesional del investigador, han permitido proponer un modelo de software que llenaría un vacío actual en la disponibilidad de herramientas para la asistencia del ingeniero de diseño. Es un paso importante que deberá ser complementado con la elaboración del software funcional.