Manejo de Requerimientos y Solución de Conflictos en Proyectos de Desarrollo de Software.

Ricardo Raúl Jacinto Montes
rjacinto@gdl.cinvestav.mx
CINVESTAV del IPN,
Unidad Guadalajara
Prol. López Mateos Sur 590,
45090, Guadalajara, Jalisco
México

Ignacio Navarro Alvarez
inavarro@gdl.cinvestav.mx
CINVESTAV del IPN,
Unidad Guadalajara
Prol. López Mateos Sur 590,
45090, Guadalajara, Jalisco
México

Javier González Sánchez
Javier@gdl.cinvestav.mx
CINVESTAV del IPN,
Unidad Guadalajara
Prol. López Mateos Sur 590,
45090, Guadalajara, Jalisco
México

Resumen.

El cambio excesivo en los requerimientos de un proyecto de software en un problema importante, para manejarlo es necesario asegurar que todos los requerimientos del proyecto han sido recolectados y es indispensable que los requerimientos sean consistentes. Para asegurar esta consistencia se propone una técnica de manejo de requerimientos que permite la identificación y posible solución de conflictos. Esta técnica funciona de manera iterativa, identificando, evaluando y solucionando conflictos hasta llegar a una solución satisfactoria. La técnica empleada para la resolución de conflictos permite encontrar soluciones inovativas que influyen en la satisfacción final de los requerimientos del cliente.

Introducción.

La calidad es un concepto difícil de definir, ya que su percepción varía de persona a persona. Debido a esta característica, la calidad de un producto sólo puede definirse con base en la satisfacción de los requerimientos del cliente para quién el producto está dirigido. El aseguramiento de esta calidad, es una de las mayores preocupaciones en el desarrollo de productos de software. Una meta que se ve obstaculizada por la excesiva transformación de los requerimientos por parte del cliente, lo que implica por un lado la casi imposibilidad de establecer una planeación estricta para el desarrollo del proyecto y por otro lado una adaptación constante de las actividades del proceso de desarrollo de software para responder a los cambios en los requerimientos del cliente.

Dada la naturaleza del desarrollo de software como un proceso diferente a cualquier otro, un proceso que normalmente implica el desarrollo de productos únicos y no repetibles, la aplicación de las técnicas convencionales de calidad no es fácilmente adaptable ya que las características del producto pueden determinar cambios en el proceso de desarrollo. Esto no implica que cada producto requiera un proceso específico, implica que existe una cierta variabilidad en el proceso de cada producto. El manejo de esta situación no es tarea fácil: por un lado es necesario asegurar que todos los requerimientos del proyecto han sido recolectados lo cuál puede abordarse por medio de guías y técnicas conocidas Booch, Jacobson o la guía de IEEE [IEEE, 1984] y por otro lado es indispensable que los requerimientos sean consistentes, por lo que se propone una técnica de manejo de requerimientos que permite la identificación de conflictos, basados en esta identificación la solución de los mismos es posible. Esta técnica funciona de manera iterativa, se identifican los conflictos, se evalúa la situación actual, si esta presenta conflictos estos se solucionan y la situación modificada sé re-evalua hasta llegar a una solución satisfactoria. La técnica empleada para la resolución de conflictos, al no estar orientada a buscar un compromiso permite encontrar soluciones inovativas que influyen en la satisfacción final de los requerimientos del cliente. El aporte de este trabajo es doble, por un lado las técnicas se adaptan al desarrollo de software con calidad y por el otro se combinan para ofrecer una metodología de administración de conflictos en los requerimientos que permite aprovechar las posibilidades de mejora en el desarrollo de software. Además la metodología resultante ha sido implementada en una herramienta que la automatiza y está disponible para su prueba en Internet: http://www.gdl.cinvestav.mx/quality.

1. Calidad en Software.

La calidad sólo puede definirse en términos de los requerimientos del usuario. Por ello una herramienta de administración de calidad debe considerar los requerimientos como su base principal. Básicamente existen dos maneras de controlar la calidad en un proceso: una a través de la prevención y la otra a través de la detección y posterior corrección. En la prevención el operador está

consciente de los requerimientos del usuario y puede corregir cualquier desviación inmediatamente. En la detección los errores se detectan por personal externo al desarrollo de proyecto, se reportan y se corrigen por el personal de desarrollo de software. El desarrollo de Software es un proceso iterativo e incremental, el cual evoluciona hasta llegar a un sistema final que satisface los requerimientos iniciales, Figura 1.

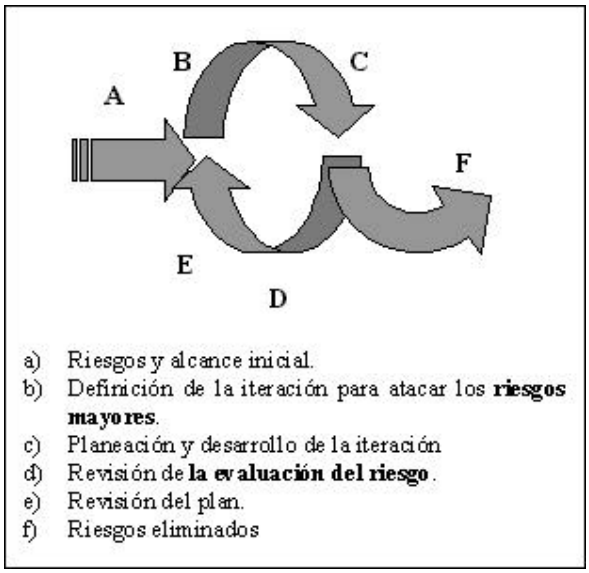

Figura 1. Ciclo de Vida del Desarrollo de Software

Este tipo de ciclo de vida busca mitigar los riesgos en el desarrollo del sistema por medio de aproximaciones y refinamientos sucesivos hasta alcanzar la meta final.

Existen en este ciclo de vida dos aspectos no cubiertos de manera específica por prácticamente ninguna metodología de desarrollo de software actualmente en uso: La definición de los riesgos mayores y la evaluación del riesgo. El tratamiento de ambos aspectos implica la definición precisa y jerárquica de la importancia de los requerimientos del cliente, la identificación de conflictos potenciales en los requerimientos del cliente y la solución de los conflictos cuando estos representan una oportunidad de mejora del sistema final. Las metodologías tradicionales sólo atacan tangencialmente estos problemas. La técnica comúnmente utilizada es la de entrevistas con el cliente, a través de las cuales se busca definir un conjunto consistente de requerimientos. El aseguramiento de la consistencia de los requerimientos es un problema cuya solución se deja a la experiencia del analista.

En el caso de las metodologías de diseño orientado objetos, estas se enfocan a la obtención de modelos computacionales del sistema real, identificando sus elementos y las relaciones entre ellos, procediendo por refinamientos sucesivos, de lo general a lo específico, para obtener así una solución computacional del problema en cuestión. Este enfoque es la base del reuso, y de la flexibilidad de las configuraciones de software actuales. Sin embargo no direcciona el problema del manejo de conflictos, de hecho podría argumentarse que la política de solución utilizada para la resolución de conflictos es el compromiso entre los diferentes factores de contradicción involucrados. Esta actitud de compromiso generalmente limita la creatividad del ingeniero, uno de las características más apreciadas en el proceso de software, cuando esta creatividad puede ser controlada. La metodología TRIZ (theory for inventive problem solving) [Terminko, 1997] permite la búsqueda metodológica de soluciones cuando los problemas se expresan por la existencia de conflictos en los elementos de la solución. Más aún, cuando TRIZ se complementa con las técnicas de análisis funcional, la aplicación de estas lleva a una identificación de clases similar a las técnicas descritas en las diferentes metodologías.

2. Metodología Propuesta

La metodología propuesta es isomorfa al ciclo de la figura 1. Implica la identificación y clasificación de los requerimientos de acuerdo a su importancia para el cliente. La visualización de esta clasificación y jerarquización implica una planeación natural, las actividades más importantes deben atacarse primero. Después se evalúa la eficiencia, esto se realiza por comparación contra un valor asignado a una solución ideal, si los riesgos no son eliminados y por ende el proyecto se encuentra lejos de la solución ideal, se efectúa una revisión de la situación actual, se busca una solución a los posibles conflictos y el ciclo se repite. La metodología propuesta es visual y permite mantener la voz del cliente en todo el proceso jerarquizando sus requerimientos, permite comparar entre sí los requerimientos del cliente y soluciones técnicas, identificar los conflictos potenciales, encontrar y evaluar soluciones a los conflictos. La metodología puede aplicarse a diferentes niveles permitiendo el manejo de los requerimientos en diferentes etapas del proceso.

La metodología se basa en dos metodologías ampliamente utilizadas en manufactura: QFD ("Quality Function Deployment") para el manejo de requerimientos y TRIZ, una búsqueda de soluciones inventivas a problemas complejos de cualquier área. QFD es una proyección conceptual de los requerimientos del cliente contra los descriptores técnicos del proyecto. Por su parte, TRIZ es un conjunto de factores de contradicción, organizados en una matriz de contradicciones. La identificación de los factores de contradicción involucrados lleva a un conjunto de principios de inventiva aplicables para encontrar la solución del problema.

TRIZ se compone además de un conjunto de técnicas que permiten encontrar soluciones inventivas para la resolución de problemas que se manifiestan por contradicción. El componente clave de TRIZ es el concepto del resultado final ideal, el cual establece que el valor ideal de una solución es aquella que reduce los daños potenciales maximizando el beneficio [Domb 1997].

En [Rovira, 1997] se describe una técnica para combinar QFD/TRIZ y análisis funcional. La metodología aquí presentada es una extensión de esta técnica al permitir una realimentación a QFD. Esta realimentación permite guiar el

proceso de mejora para conciliar los descriptores técnicos con los requerimientos del cliente. La metodología se compone de dos fases iterativas: una de manejo de requerimientos y otra de manejo de conflictos, dentro de esta fase se encuentra una actividad llamada análisis funcional (FA). La pertinencia de esta actividad se justifica por su aplicabilidad a las metodologías tradicionales de desarrollo de software orientado objetos. Sin embargo, el FA utilizado no sigue el enfoque de [Rovira, 1997] de descomposición funcional en subfunciones, este enfoque ha sido abandonado por la comunidad de computación porque tiende a llevar a soluciones demasiado específicas poco flexibles, dificultando el reuso de componentes. En su lugar el FA es utilizado exclusivamente en el nivel de abstracción de trabajo. Este aspecto es algo inherente a la metodología, QFD obliga de manera natural a restringirse a un nivel determinado de abstracción de manera paralela al proceso de desarrollo OO, cuyo principio de base es el del "mínimo compromiso", no comprometerse a ninguna decisión de diseño o implementación hasta que sea absolutamente necesario.

Manejo de requerimientos

El manejo de requerimientos comienza después de la fase de recolección y se refiere al análisis de los mismos y como se mencionó se basa en QFD. QFD pone el análisis de la calidad al frente de la administración del proyecto, cuantifica los parámetros de calidad y producción desde el inicio del proceso de desarrollo, estableciendo una firme identidad del producto. QFD requiere de un trabajo concurrente, en cada fase del proceso de desarrollo. QFD propone cuatro fases para convertir los requerimientos del cliente en planes de producción: planeación del producto (casa de la calidad), diseño del producto, planeación del proceso y control del proceso. Para este trabajo sólo las dos primeras fases son relevantes. La fase planeación del producto en QFD establece la construcción de la casa de la calidad donde se representan los parámetros de planeación de manera visual. La viabilidad de la casa de la calidad se confirma si analizamos el proceso de conformación de un equipo de desarrollo y sus primeros pasos. El primer paso de un equipo de desarrollo en un proyecto es asegurarse de la misión y alcance del proyecto por parte del equipo.

La casa de la calidad (HoQ) es una manera de considerar simultáneamente las necesidades del cliente, las posibilidades técnicas de la empresa y el comportamiento de la competencia en proyectos similares. QFD es una proyección de diferentes aspectos del proyecto para la visualización de sus interrelaciones. Esta visualización es posible utilizando la matriz de la figura 2 que comprende cuatro zonas principales: correlación, factores técnicos, requerimientos y competencia. Las zonas de requerimientos, factores técnicos y competencia permiten la clasificación de sus elementos de acuerdo a valores que reflejan la importancia del elemento en el producto final. La zona de Correlación pone en evidencia las interrelaciones entre los elementos de la zona de requerimientos y la zona de factores técnicos. Utilizar HoQ implica la recolección de los requerimientos del cliente, un estudio de mercado para evaluar los productos de la competencia y la evaluación de los elementos técnicos del proyecto.

El orden en que debe ser llenada la casa de la calidad es el que tiene en la figura 2, la zona de correlación es la que se explica por ser la más relevante, en [Becker 1997; Hales 1997; Lamia 1997] se encuentra información de los demás pasos de la HoQ, Con el objetivo de compatibilidad en la nomenclatura, se emplean las voces anglosajonas WHAT por QUE y HOW por COMO.

Correlación requerimiento-descriptor. Relacionar los deseos del cliente con las habilidades de la empresa para dimensionar adecuadamente el esfuerzo e identificar las actividades que requieren mayor atención es un aspecto importante en la planeación adecuada del proyecto. Esta correlación permite identificar las fortalezas y debilidades del producto.

Importancia absoluta de los descriptores técnicos. La importancia absoluta de cada descriptor técnico es la

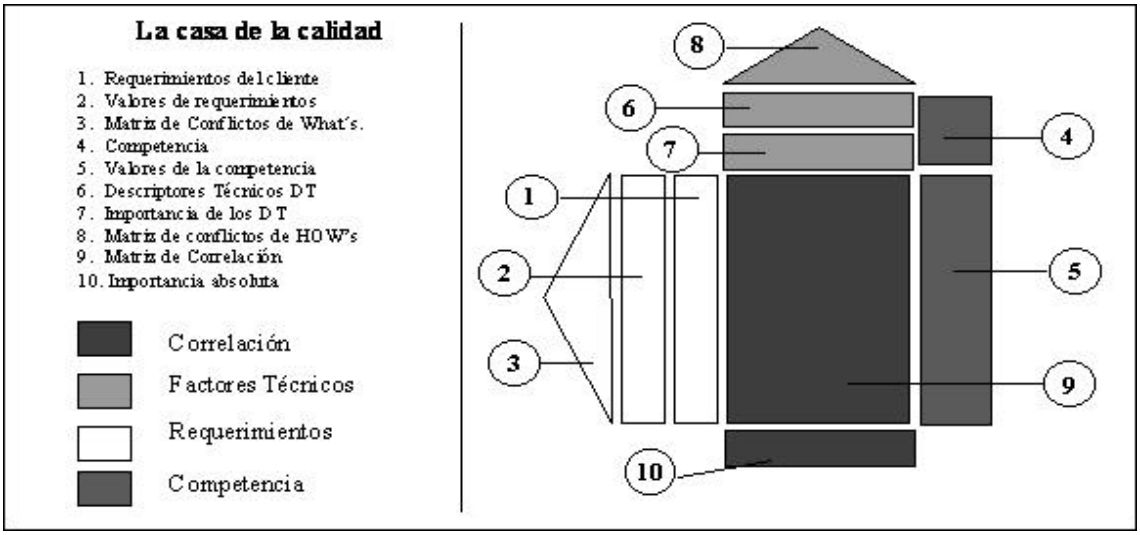

Figura 2. Casa de la Calidad, HoQ

relevancia del descriptor dentro del desarrollo del proyecto. Es una combinación de los requerimientos del cliente y de su importancia técnica, que permite al equipo de trabajo definir sus prioridades de trabajo. El cálculo numérico es el producto del valor de la celda y la clasificación de importancia del cliente, los números son luego sumados en sus columnas respectivas. El algoritmo de cálculo para la importancia absoluta está definido de la siguiente manera: Sea t_i el valor de la importancia técnica para el HOW_i. Sea h_i el valor que representa el costo relativo de implementar el descriptor (HOW). Sea $r_{i,j}$ el valor que representa el valor de correlación entre el HOW_i y el $WHAT_j$.

Para todo i, donde $0 < i < n + 1$, y n es igual al número de elementos HOW tenemos que $t_i = \Sigma_j h_i * r_{i,j}$. Con $0 < j < m + 1$, m es el número de elementos WHAT.

En esta primera fase el esfuerzo se ha concentrado en la comprensión del problema y en la identificación de conflictos potenciales que puedan afectar el desarrollo del proyecto. La segunda fase de la metodología propuesta introduce el análisis de contradicciones para requerimientos o necesidades técnicas que a pesar de ser necesarias para cumplir un cierto aspecto del producto entran en conflicto con otro.

Manejo de Conflictos

La cobertura de la identificación de conflictos por medio de QFD, descrito anteriormente puede no ser completa, por lo que la aplicación de técnicas complementarias es obligatoria. Un método alternativo es el Análisis Funcional (FA) aplicado a los descriptores técnicos considerados como peligros potenciales por QFD. En desarrollos de software orientado a objetos, los descriptores técnicos son equivalentes a clases, por lo que el uso de esta metodología complementa el proceso natural de desarrollo de software.

Análisis funcional sobre los descriptores técnicos. Una vez obtenida la pirámide que establece la relación entre los descriptores técnicos (el techo de la casa de calidad), se analiza su relevancia en el sistema por la evaluación de la capacidad de cada descriptor técnico para resolver los requerimientos del cliente. El método de análisis se describe en la Tabla 1.

Declaración de Función			Análisis	Depurar	
A	Efectúa esto a	B	Útil o Dañino	¿Es Necesaria la Función?	¿Puede B Hacerlo? ¿Como?

Tabla 1: Análisis de funciones

En una experiencia no controlada, esta tabla se utilizó como mecanismo de descubrimiento de clases con resultados comparables o mejores que las técnicas tradicionales de descubrimiento de clases descritas en metodologías de diseño orientado a objetos.

TRIZ y el análisis de contradicciones. La teoría de la solución inventiva de problemas, que es la base de TRIZ, considera como hipótesis que las contradicciones pueden ser resueltas. TRIZ es una herramienta poderosa como auxiliar en la solución de problemas con dificultad técnica que requiere inventiva, esto es, problemas donde una o más contradicciones técnicas están envueltas y las cuales no tienen un medio conocido de solución. TRIZ se fundamenta en los siguientes pilares: El diseño ideal es la meta, las contradicciones ayudan a solucionar problemas y el proceso de innovación puede ser estructurado sistemáticamente (la inspiración no es necesariamente un proceso aleatorio).

Una vez que un problema se estructura como una contradicción, existen métodos para resolver tales contradicciones. Los problemas de este tipo se clasifican como problemas inventivos. TRIZ se estructura alrededor de PARAMETROS de CONTRADICCION organizados en una matriz [Domb, 1998], el cruce de dos parámetros define una zona de conflicto, al cual es posible aplicar un conjunto de PRINCIPIOS de INVENCIÓN [Tate, 1997]. La aplicación de los principios de invención se efectúa siguiendo un conjunto de pasos conocidos como ARIZ [Marconi, 1984]. La relación entre QFD y TRIZ se da entre los descriptores técnicos involucrados en un conflicto en la HoQ y los parámetros de contradicción de TRIZ. En la parte siguiente se describe el proceso de solución de conflictos establecido por TRIZ.

Parámetros de contradicción. Los parámetros de contradicción resumen los problemas más comunes encontrados en el desarrollo de un producto de software, son un subconjunto de 23 factores de contradicción adaptados a las características de proyectos de software.

Principios de invención. Los principios de inventiva, ofrecen soluciones a las contradicciones, se trata de 26 principios aplicables de manera general y donde en cada uno de ellos se engloban soluciones que resultaron útiles en un proyecto en particular. A su vez entre los propios principios de inventiva existen prioridades, principios que son contrarios a otros y que por tanto no pueden ser aplicados al mismo problema.

Proyección de inventivas a soluciones. Es posible clasificar los conflictos en una categoría definida, esto es proyectar los conflictos detectados con los parámetros de contradicción descritos anteriormente. Dado que se conoce una serie de elementos (parámetros) que causan problemas en la implantación y desarrollo de sistemas y a su vez se cuenta con una relación de que principios se pueden emplear para resolverlos, se obtiene una relación contradicción-inventiva, que permite identificar las acciones que deben realizarse con el proyecto en desarrollo.

3. Aseguramiento de la calidad en software.

CMM reconoce la necesidad de la administración de calidad a partir del 2º nivel. ISO9000-3 [Kehoe, 1995] requiere la presencia de esta función como requisito para alcanzar la certificación. Ambas normatividades reconocen como necesaria esta función, pero ninguna de ellas especifica la

manera de efectuarla. La herramienta propuesta ayuda a iniciar la etapa de análisis y recolección de requerimientos para un sistema de software, pero deja abierta la posibilidad de interactuar con otras metodologías en la cuestión de "analizar el problema". La metodología propuesta no es necesariamente lineal, existen ciertos pasos que pueden realizarse de manera concurrente lo que aumenta la flexibilidad de la metodología, figura 3.

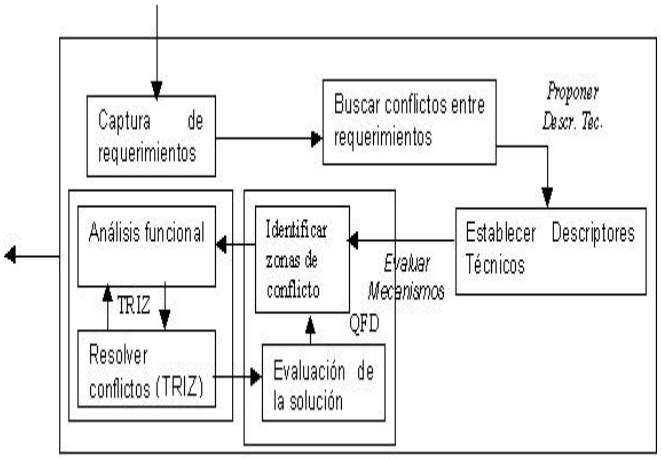

Figura 3. Fases de la metodología

Evaluación de la solución. Encontrar una solución satisfactoria requiere de una evaluación de la misma para determinar si nos proporciona el grado de satisfacción buscado. Una evaluación debe ser lo más objetiva posible (de preferencia cuantitativa). El método de evaluación mide la diferencia entre el valor ideal y el porcentaje de avance logrado gracias a la aplicación de la metodología.

Método de seguimiento de la calidad. El objetivo de formular el resultado final ideal(IFR) [Domb, 1997] es eliminar el trabajo duplicado (resolver el problema la primera vez). IFR ayuda a alcanzar avances pensando en la solución, no en los problemas que intervienen. El principio básico de TRIZ es que los sistemas evolucionan con el incremento de la idealidad(IA), donde idealidad esta definida como:

IA=Suma_de_beneficios/ (Suma_de_Costos + Suma_de_perdidas)

Donde IA, es la idealidad actual obtenida. La evolución esta en la dirección de incrementar los beneficios, decrementar los costos y eliminar las pérdidas. De tal modo que el IFR tiene todos los beneficios, ninguna pérdida y ningún costo extra al presupuestado para el problema original. El IFR tiene elimina deficiencias y preserva las ventajas del sistema original, no hace el sistema mas complicado (usa recursos gratis o disponibles) y no introduce nuevas desventajas.

Para describir el método que busca la obtención del IFR es necesario definir los parámetros de su formula.

La *suma de beneficios* es el valor obtenido de sumar todos los valores de los WHATS que no tengan ningún conflicto entre sí, esto refleja el beneficio actual obtenido.

La *suma de costos* es el valor obtenido de sumar todos los valores asignados a los HOWS.

La *suma de pérdidas* se obtiene a partir de la suma de todos los conflictos registrados tanto en HOWS como en WHATS. Para obtener el valor de un conflicto se aplica la siguiente fórmula: Cx = (WHy + WHz) / 2

Donde Cx representa el conflicto generado entre WHy y WHz, y WH representa el valor de un HOWS o un WHATS. La suma de pérdidas esta definida por $\Sigma\ Cx_i$, donde $0 \leq i \leq n$, n es el número máximo de conflictos que pueden existir. La formula que permite obtener el valor del IFR ideal específico al proyecto es:

IFR = Suma_de_beneficios / Suma_de_Costos.

A partir del valor IA e IFR, se obtiene un porcentaje de avance (PAC) en el mejoramiento de la calidad del proyecto utilizando la siguiente formula: PAC = (IA / IFR) * 100

4. Ejemplo de uso: Catalogos de Información.

En esta sección se presenta el uso de la herramienta para el análisis de un sistema de información en red.

Requerimientos del sistema. El primer paso es establecer la lista de requerimientos del sistema que aparece del lado izquierdo en la matriz "Casa de la Calidad" , igualmente se muestran los valores dados por el cliente de la importancia de cada requerimiento (una escala de 0.0 a 5.0 real).

Descriptores técnicos. A los descriptores técnicos también se les otorgó un valor numérico en la misma escala (0.0 a 5.0 con valores reales), dichos valores establecen su grado de complejidad, a mayor valor mayor complejidad.

Análisis de la competencia. Como resultado del análisis de productos similares en el mercado se definieron los valores dados por la competencia a cada requerimiento.

Identificación de conflictos. Por ejemplo en la HoQ de Ergonomía y Funcionalidad, se detecta el siguiente conflicto aplicando el Análisis Funcional de la Tabla 2.

No.	Descriptor técnico	Complejidad
1	Sistema Operativo	1.0
4	Lenguaje de Desarrollo	4.0

Declaración de Función			**Análisis**	**Depurar**	
A	Efectúa a	B	Util Dañino	Necesaria la Función	¿Puede B Hacerlo? ¿Como?
4	Depende del sistema operativo	1	Dañino	Sí	No

Tabla 2.

La aplicación del FA permite detectar el problema de que el lenguaje de desarrollo es dependiente del sistema operativo, lo que implica que el sistema no se puede ejecutar en diferentes sistemas operativos.

Primer calculo de seguimiento de la calidad.

El sistema muestra el valor de idealidad actual, figura 4, con los siguientes parámetros:

IA= 10.0/(13.0+10.0) = 0.4347
IFR= 10.0/13.0 = 0.7692
PAC = (0.4347/0.7692)*100 = 56.51%

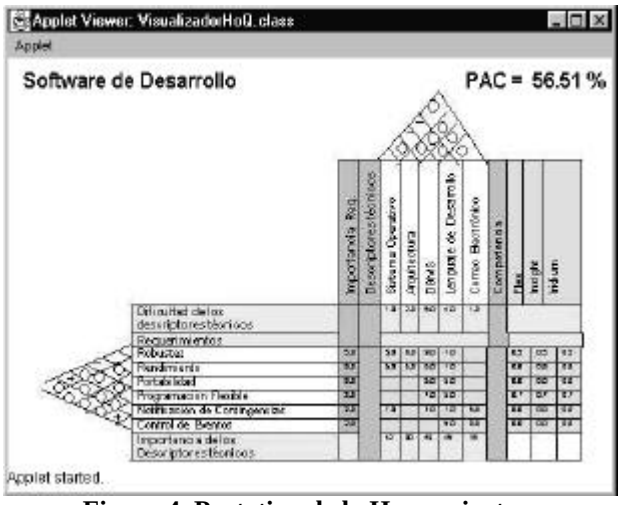

Figura 4. Prototipo de la Herramienta

Proyección del conflicto con algún parámetro de contradicción. Los parámetros de contradicción que se detectan son Portabilidad, Mantenimiento, Usabilidad y Facilidad de programación.

Búsqueda de los principios de inventiva adecuados para la solución del conflicto. De los principios de inventiva que se relacionan con el conflicto encontramos que el que más se ajusta al problema es el principio de "*Intermediación*", que recomienda utilizar un elemento o proceso intermediario.
Ahora se busca en la base de conocimientos de la herramienta todas las soluciones agrupadas dentro de este principio, supóngase que ninguna le satisface, entonces tiene la posibilidad de proponer una nueva solución y agregarla a la base de conocimientos, por ejemplo una solución a la medida, seria "El uso de un lenguaje interpretado por una maquina virtual independiente del sistema operativo", de tal forma que todos las aplicaciones desarrolladas para el sistema serian independientes de la plataforma.
Finalmente resuelto el conflicto el sistema se calcula otra vez el porcentaje de idealidad:
IA= 10.0/(13.0+5.0) = 0.5555
IFR= 10.0/13.0 = 0.7692
PAC = (0.5555/0.7692)*100 = 72.21 %
Lo que implica que existe un mejoramiento en la calidad del sistema respecto a los requerimientos del cliente.

Importancia Técnica. Entre mayor el valor, mayor la importancia del descriptor técnico. En este caso el DBMS es el de mayor importancia al tener un valor de 49.

Herramienta. Esta metodología ha sido automatizada en un prototipo lo que permite una mayor flexibilidad de uso. La herramienta busca mejorar la búsqueda de soluciones y permite generar una base de casos que ayuden a guiar esta búsqueda. Esta implementada en Java y CGI, lo que permite una independencia de plataforma al poder ser ejecutada en un navegador, figura 4.

Conclusiones y trabajo futuro.

La metodología presentada permite obtener un conjunto de requerimientos consistentes que es la base de un desarrollo correcto. Más aún, el manejo de conflictos utilizado permite la aplicación metódica de un conjunto de principios que facilitan el encontrar soluciones innovadoras a los problemas encontrados durante la fase de análisis de requerimientos. Las características de esta metodología complementan a las metodologías tradicionales de desarrollo de software, ya que atacan problemas normalmente no contemplados obteniendo soluciones que permiten aprovechar los conflictos para mejorar el producto. Actualmente se trabaja en la validación de esta metodología aplicándola a la resolución de casos reales y así como lograr que la herramienta automatice por completo la metodología.

Referencias.

[IEEE, 1984] Std. 830-1984, IEEE Guide to Software Requirements Specifications.
[Becker, 1997] The House of Quality. Becker Associates Product Planning, 1997.
[Dean, 1997] Dean E. B., Seven Management (New) Tools from the Perspective of Competitive Advantage, 1997.
[Domb, 1998] Domb, E., The 39 features of Altshuller's contradiction matrix, novembre 1998.
[Domb, 1997] Domb, E., "The Ideal Final Result: Tutorial", 1997.
[Hales, 1997] Hales, R. F., QFD and the Expanded House of Quality. ProAction Development, Inc, 1997.
[Herzwurm, 1997] Herzwurm, G., W. Mellis, S. Schockerts and C. Weinberger, Customer Oriented Evaluation of QFD Software Tools. University of Cologne, Chair in Business Computing and QFD Institut Deutschland, 1997.
[Kehoe, 1995] Kehoe, R. and A. Jarvis, ISO 9000-3, A Tool for Software Product and Process Improvement, Ed. Springer 1995, ISBN-0-387-94568-7.
[Lamia, 1997] Lamia, W. M., Integrating QFD with Object Oriented Software Design Methodologies. Software Engineering Institute, Carnegie Mellon University, 1997.
[Marconi, 1984] Marconi, J., "ARIZ: The algorithm for inventive problem solving", 1984.
[Rovira, 1997] Rovira, N. L. and H. Aguayo, A new Model of the Conceptual Design Process using QFD/FA/TRIZ. Instituto Tecnologico de Estudios Superiores de Monterrey, 1997.
[Teminko, 1997] Teminko, J., The QFD, TRIZ and Taguchi connection: customer-driven robust inovation, Ideation International Inc. The Ninth Symposium on Quality Function Deployment, June 1997.
[Tate, 1997] Tate, K. and E. Domb, The 40 inventive principles and the accompanying examples, May 1997.