Aseguramiento de la Calidad en la Administración de Proyectos de Desarrollo de Software.

Manejo de Requerimientos y Solución de Conflictos.

I. Navarro, J González, R, Jacinto, F. Ramons
CINVESTAV del IPN, Unidad Guadalajara
{Inavarro, javier, rjacinto, framos} @gdl.cinvestav.mx

R. Gómez
ITESM-CEM
rogomez@campus.cem.itesm.mx

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 la satisfacción de los requerimientos de un cliente, y por lo tanto el aseguramiento de calidad, es una de las mayores preocupaciones en el desarrollo de productos de software.

El aseguramiento de calidad es una actividad bien entendida en el marco de procesos de manufactura. En donde la aplicación de técnicas estadísticas [Gordon, 1995] para la medición de desviaciones en los procesos de producción es una práctica cotidiana. Estas técnicas resultan bien adaptadas para procesos bien definidos y repetitivos con un conjunto de requerimientos estables en el tiempo, enfocados al desarrollo de productos en serie.

Dada la naturaleza del desarrollo de software, el cual normalmente se maneja por proyectos lo que implica el desarrollo de productos únicos y no repetibles, la aplicación de este tipo de técnicas 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 lo que dificulta la medición estadística de desviaciones en el proceso.

Uno de los argumentos más socorridos para explicar la permanente “Crisis del Software” en la que parecen estar todos los proyectos de software es 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.

El manejo de esta situación es una obligación de un buen líder de proyecto. Sin embargo, esta no es tarea fácil: por un lado es necesario asegurar que todos los requerimientos del proyecto han sido recolectados y por otro lado es indispensable que los requerimientos sean consistentes. La primera tarea puede abordarse por medio de la identificación de casos de uso [Booch, Jacobson] o utilizando la guía de estándares como los del IEEE[ANSI/IEEE, 1984]. La segunda tarea es el objetivo de este trabajo, en este artículo se propone una técnica de manejo de requerimientos que permite la identificación de conflictos, con base 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 se reevalua 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 innovativas que influyen en la satisfacción final de los requerimientos del cliente.

La técnica desarrollada además ha sido implementada en una herramienta computarizada que permite la automatización del proceso. La técnica se basa para el manejo de requerimientos (administración, jerarquización, identificación de conflictos) en la metodología QFD (Quality Function Deployment) y para la búsqueda de soluciones inventivas en TRIZ (acrónimo ruso para Teoría para la Resolución de Problemas Inventivos) ambas técnicas son actualmente ampliamente utilizadas en la industria de la manufactura. 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. La metodología resultante ha sido implementada en una herramienta que la automatiza y está disponible para su prueba en http://www.gdl.cinvestav.mx/quality.

Este artículo se estructura de la siguiente manera, en la primera sección titulada "Calidad en Software" introduce las implicaciones del aseguramiento de la calidad en el proceso de desarrollo de Software y el proceso ideal para llegar a un sistema final que satisfaga los requerimientos originales del cliente. En la sección tres se trata la "Metodología Propuesta", una metodología isomorfa al ciclo de desarrollo, implicaciones y requerimientos.

La sección cuatro "Manejo de Requerimientos" trata a fondo la aplicación de la primera fase de la metodología que abarca la etapa de aplicación de la matriz "La Casa de la Calidad". Dado que esta no permite la total cobertura del manejo de conflictos, se proponen técnicas complementarias como el Análisis Funcional y su análisis posterior con el análisis de contradicciones, esto en la sección cinco "Metodología de Aseguramiento de la Calidad" que engloba todas las técnicas aisladas que se contemplaron en las secciones previas y formula un nuevo planteamiento. Por ultimo en la sección seis se propone un "Ejemplo de Aplicación" que utiliza la metodología propuesta, el cuál analiza una herramienta CASE.

2. Calidad en Software.

La Calidad en el PDS.

El aseguramiento de calidad en el PDS implica respetar los requerimientos del cliente. La calidad sólo puede definirse a partir de los requerimientos del usuario, por lo que 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:

1) Prevención.
2) Detección y 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. Ambos enfoques son complementarios, la realización de un trabajo buscando evitar los errores desde el principio es una filosofía que debe ser parte del trabajo diario de una organización. La prevención de errores implica la conciencia de que los errores son difíciles sino imposibles de evitar, detectarlos y corregirlos es parte de todo proceso enfocado a la calidad.

El Proceso 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 (Fig. 1). Este tipo de ciclo de vida busca mitigar los riesgos en el desarrollo del sistema por medio de aproximaciones y refinamientos sucesivos para alcanzar la meta final.

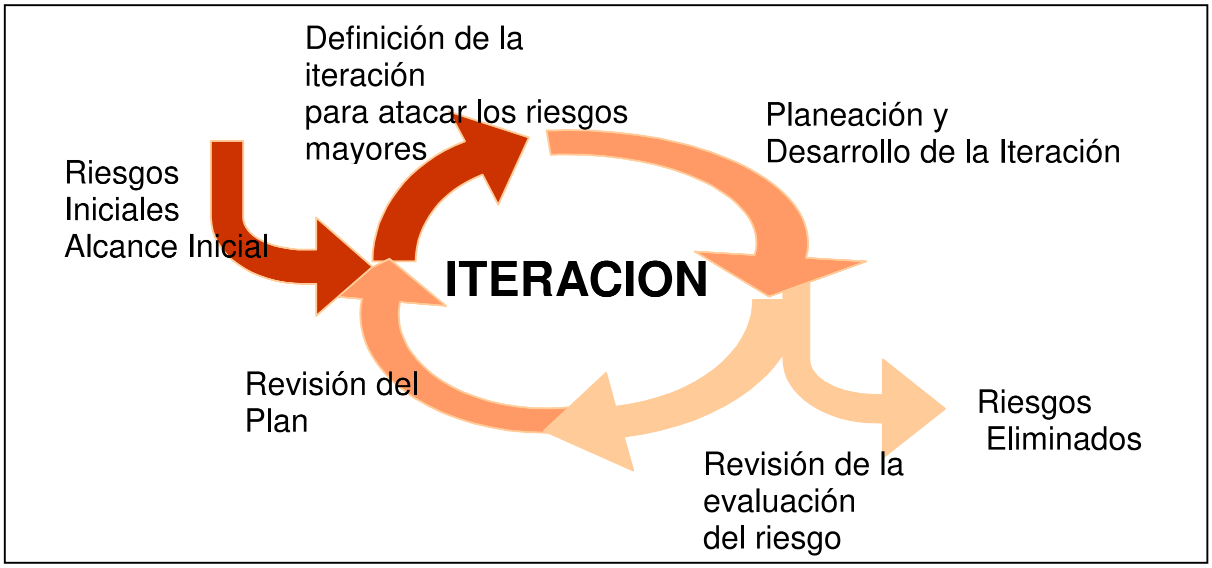


Figura 1. Ciclo de desarrollo de Software

Existen en este ciclo de vida dos aspectos actualmente no cubiertos de manera específica por prácticamente ninguna de la 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 de la jerarquía de 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. Una revisión (no exhaustiva) del material de Ingeniería de Software no permitió identificar una metodología o actividades encaminadas a la jerarquización de requerimientos, o la identificación y solución de conflictos.

En general todas las metodologías asumen que el análisis del sistema incluye los aspectos mencionados, sin embargo no existe la especificación de las actividades necesarias para cubrir estos aspectos. Las causas pueden explicarse por el desarrollo de la Ingeniería de Software, por el punto de vista particular de la metodología utilizada o por otras razones. Por ejemplo en Orientado Objetos OO el punto de vista es la identificación de los elementos del sistema que realizan funciones y sus relaciones. Las metodologías OO (Rational, FUSION, OMT,...) se basan en este punto de vista, buscando reducir los riesgos por la aplicación del ciclo de vida de la figura 1. Los riesgos se mitigan por la búsqueda incremental de soluciones que eventualmente convergen a la solución final que soluciona el conjunto de requerimientos del cliente.

El análisis específico de los requerimientos para jerarquizarlos por su importancia, la identificación de los conflictos potenciales y su solución no debe verse como un elemento contradictorio a las metodologías tradicionales sino como una actividad complementaria para mejorar el proceso de desarrollo de software. En efecto el contar con actividades especificas y técnicas apropiadas con criterios definidos solo puede redundar en una disminución del riesgo en cada iteración, lo que debe implicar una aceleración en el proceso normal de desarrollo de software.

Límites del Diseño Orientado a Objetos

Las metodologías de diseño orientado objetos 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 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 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[1].

Metodología Propuesta

La metodología propuesta es isomorfa al ciclo de vida 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. La evaluación de la 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 los requerimientos del cliente y compararla con los requerimientos y soluciones técnicas, identificar los conflictos potenciales, encontrar soluciones a los conflictos y evaluar las soluciones obtenidas. La metodología puede aplicarse a diferentes niveles permitiendo el manejo de los requerimientos en diferentes etapas del proceso.

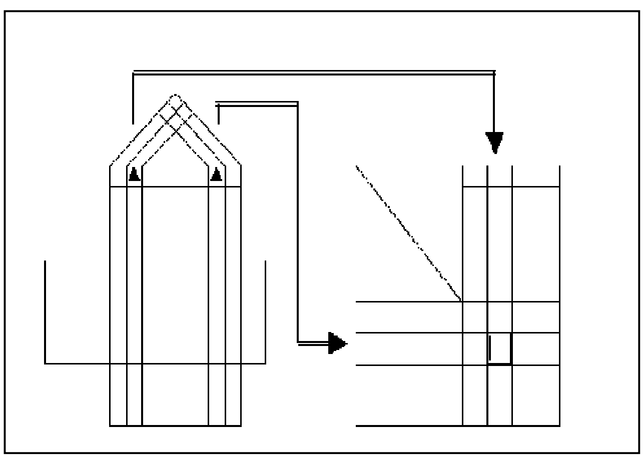


Figura 2. Relación entre QFD y TRIZ

Esta 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

[1] Búsqueda de sustantivos y proponerlos como clases candidatas, procedimientos asociados a estos, aplicación de los procedimientos a otros candidatos, identificación de los procedimientos aplicados a la clase candidata, etc.

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[2]. 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. La idealidad puede expresarse como:

IA = Suma_de_beneficios / (Suma_de_Costos + Suma_de_pérdidas) (FID)

En [rovira] 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] 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. En la metodología este se basa en Quality Function Deployment (QFD) la cual provee un bosquejo para producir productos de mejor calidad al menor costo posible, al unir al productor y al consumidor desde las primeras etapas del proceso de producción. Es una herramienta de uso conjunto la cual captura los requerimientos del cliente y transforma esas necesidades en características de un producto o servicio. 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 (La casa de la calidad).
- Diseño del producto.
- Planeación del proceso
- Control del proceso

Para este trabajo sólo las dos primeras fases son relevantes para su aplicación al proceso de desarrollo de software. 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.

[2] Originalmente TRIZ identifica 40 factores de contradicción, el análisis efectuado revela que sólo 26 son aplicables a software.

1. Requerimientos del cliente
2. Valores de requerimientos
3. Matriz de Conflictos de What´s.
4. Competencia
5. Valores de la competencia
6. Descriptores Técnicos DT
7. Importancia de los DT
8. Matriz de conflictos de HOW's
9. Matriz de Correlación
10. Importancia absoluta

Correlación

Factores Técnicos

Requerimientos

Competencia

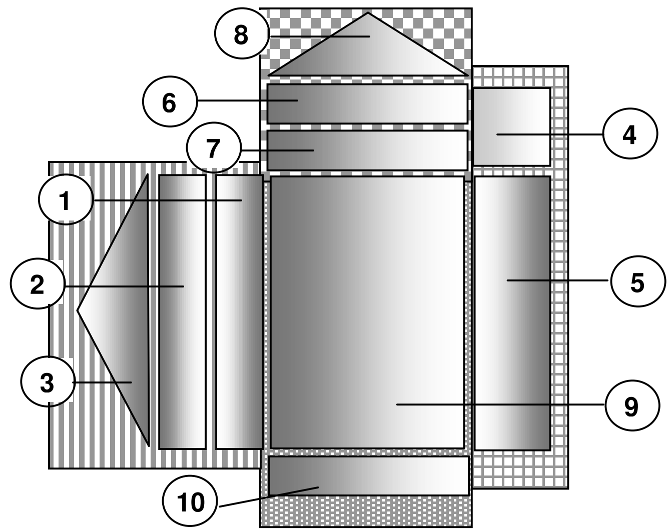


Figura 3. La casa de la calidad

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 3 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. Cada uno de estos aspectos requiere de un método y procedimientos adecuados los cuales se describen no exhaustivamente a continuación, siguiendo el orden de la figura 3.

REQUERIMIENTOS

1. Captura de los requerimientos del cliente.

Después de la etapa de recolección de requerimientos propia de cualquier proyecto, se procede a la ubicación y clasificación de los requerimientos, respondiendo al conjunto siguiente de preguntas llamado "cinco ¿*WHAT*? y un ¿*HOW*?"[3]:

- ¿ Quién tiene el problema?
- ¿ Que parece ser el problema?
- ¿ Cuándo ocurre el problema?, ¿Todo el Tiempo?, ¿Bajo ciertas circunstancias?
- ¿ Dónde ocurre el problema?
- ¿ Porque ocurre el problema?
- ¿ Cómo ocurre el problema?

[3] Con el objetivo de compatibilidad con la nomenclatura, en este trabajo se emplean las voces anglosajonas WHAT por QUE y HOW por COMO.

Este conjunto nos puede conducir a la identificación de la zona de interacción que es la fuente del problema, y tiene el beneficio de familiaridad de técnica y vocabulario. Esta fase también considera la documentación de requerimientos fijados por estándares aplicables.

2. Asignación de valores de importancia a los requerimientos según el cliente.

Establecer la importancia del requerimiento por el cliente. La importancia se expresa por medio de un valor en el rango de 0.0 a 5.0, con 5.0 correspondiendo a la mayor importancia del requerimiento. La manera de establecer esta importancia puede ser, por ejemplo, por medio de la aplicación de algunas técnicas conocidas en los círculos de calidad: lluvia de ideas, votación, pareto.

3. Matriz de relación WHAT vs WHAT.

Descubrir los conflictos en los requerimientos del cliente lo más temprano posible permite la resolución de problemas desde su raíz. Este proceso de descubrimiento requiere la participación del cliente y del proveedor, para asegurar que ambos comparten la misma visión del proyecto. Igualmente, en este proceso, es posible la aplicación de técnicas de círculos de calidad que ayuden a la identificación de estos conflictos.

COMPETENCIA

4. Clasificación de la competencia según el cliente.

Entender como la competencia resuelve los requerimientos del cliente puede ser un tremendo avance competitivo, por lo que en este paso se agregan los nombres de todos los competidores que tienen algún producto o servicio similar al que se va a desarrollar.

5. Análisis técnico de los productos de la competencia.

Por cada requerimiento del cliente considerado por el competidor se asigna un valor (0.0 a 5.0) a cada requerimiento del cliente, entre más grande es el valor, mejor resuelve la competencia cierto requerimiento.
Esta fase es opcional, su adopción puede ayudar en el caso de desarrollo de productos genéricos y nos proporciona una idea de cómo esta posicionado nuestro producto frente a la competencia.

DESCRIPTORES TÉCNICOS

6. Establecimiento de los Descriptores Técnicos.

Un descriptor técnico es la descripción del elemento de ingeniería que soluciona un requerimiento del producto. Un producto o proceso puede ser descrito en términos de sus partes y en términos de sus descriptores técnicos. Así que la expresión "De la forma sigue el descriptor técnico", significa que cada forma (producto o proceso) es una expresión concreta de uno o más descriptores técnicos. Esta parte del proceso implica considerar desde el inicio La voz del Ingeniero.
Una descripción completa de un descriptor técnico consta de tres partes: Un sujeto, un verbo y un objeto, por ejemplo un auto es una forma que satisface el cómo "Transportar gente". Para expresar lo anterior definimos al "Auto" como sujeto, "Transporta" como verbo y "Gente" como objeto. Es de notar la correspondencia con el proceso de descubrimiento de clases en un sistema orientado a objeto. La identificación de clases potenciales se hace identificando el sustantivo (el objeto) y las acciones de las que es capaz y sobre el objeto que actúa [OMT, BOOCH].

7. Valores ideales de los descriptores técnicos.

Estos valores representan el costo subjetivo para los descriptores técnicos. El costo de los descriptores técnicos que el Ingeniero asigna de manera subjetiva debe tomar en cuenta el costo de los materiales, mano de obra, manejo, transportación, procesamiento y el costo de hacer el negocio si alguna de estas partes son compradas a un proveedor o producidas por el usuario. Lo anterior se puede establecer en un valor real dentro del rango de 0.0 a 5.0. El éxito en los negocios significa incrementar el valor ofrecido a usuarios y clientes. Así que observamos dos formas de incrementar el valor mencionado:

1. Incrementar el número y confiabilidad de los mecanismos.
2. Decrementar el número de partes, o el costo de las partes individuales.

8. Matriz de relación HOW vs HOW.

Analizar las interacciones entre descriptores técnicos para identificar conflictos potenciales, implica por un lado evitar trabajo innecesario y por otro lado la oportunidad de mejora del producto. Los miembros del equipo deben examinar de que manera cada descriptor técnico impacta a los demás, documentar las relaciones fuertemente negativas entre descriptores técnicos (identificación de los conflictos).
Esta actividad solamente implica la jerarquización de los descriptores técnicos y la identificación de conflictos potenciales, el manejo de los mismos (oportunidad de mejora) se trata en la siguiente sección de este trabajo.

RESULTADOS

9. Matriz de correlación.

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. Es importante remarcar que este trabajo permite el desarrollo interdisciplinario de un producto, ya que no existe una limitación sobre los descriptores técnicos, los cuales pueden cubrir no sólo el software sino también el hardware y aspectos como los de la Interfaz hombre-máquina.

10. Importancia absoluta de los descriptores técnicos.

La importancia absoluta de cada descriptor técnico es la 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 calculo 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 numero de elementos HOW.
  $t_i = \Sigma_j h_i * r_{i,j}$

Con $0 < j < m + 1$, m es el número de elementos WHAT.

La importancia de esta información radica en que facilita la toma de decisiones respecto a la importancia de las actividades del proyecto, ya que proporciona una guía para el control del avance del mismo y la definición de plazos y presupuestos.

La documentación de los requerimientos del cliente, de los descriptores técnicos y de la habilidad técnica de la organización para cumplir con tales requerimientos es crucial para pasar a la siguiente fase: el manejo de conflictos, en la cual se requiere creatividad. En esta 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.

5. Manejo de Conflictos

Identificación de conflictos.

La cobertura de la identificación de conflictos por medio de QFD 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 OO, 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 mecanismos.

Una vez obtenida la pirámide que establece la relación entre los descriptores técnicos (el techo de la HoQ), se analiza su relevancia en el sistema por la evaluación de la capacidad de cada descriptor técnico a resolver los requerimientos del cliente. Es decir se analiza la funcionalidad de cada descriptor técnico con respecto al sistema global. El método de análisis se sumariza en la siguiente tabla []:

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

Tabla análisis de funciones

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

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 fundamente en los siguientes pilares:

1. El diseño ideal es la meta.
2. Las contradicciones ayudan a solucionar problemas
3. 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, el cruce de dos parámetros define una zona de conflicto, al cual es posible aplicar un conjunto de PRINCIPIOS de INVENCIÓN. La aplicación de los principios de invención se efectúa siguiendo un conjunto de pasos conocidos como ARIZ.

La relación entre QFD y TRIZ está dada por el mapeo (figura 2) 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 los 40 factores de contradicción adaptados a las características de proyectos de software , el conjunto de los factores se reporta en la tabla 1 del anexo A.

Principios de invención.

Los principios de inventiva, ofrecen soluciones a las contradicciones, se trata de principios aplicables de manera general y donde en cada uno de ellos se engloban soluciones que resultaron útiles en un proyecto en particular, la tabla descriptiva de los principios de inventiva se localiza en el anexo B.

Proyección de inventivas a soluciones.

La identificación de los conflictos existentes entre los descriptores técnicos, a través de HoQ de QFD, 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 que se pueden emplear para resolverlos, se obtiene una relación contradicción-inventiva, donde la idea es proyectar los conflictos de los descriptores técnicos frente a las contradicciones y de esa manera obtener una relación inventiva-solución que nos lleve a identificar las acciones que deban realizarse con el proyecto en desarrollo.

Se cuenta con grupos definidos de conflictos denominados "parámetros de contradicción" (23 parámetros identificados como prioritarios). Cualquier conflicto que pueda identificarse entre dos descriptores técnicos cae irremediablemente en una de esas categorías (problemas de espacio, robustez, portabilidad, fiabilidad, etc.).

Para solucionar a cada uno de esos conflictos se encuentran 26 principios de inventiva que son aplicables con éxito a ciertos problemas de desarrollo de software; cada uno aplicable en un grado de éxito a ciertos parámetros de contradicción. 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 Todos estos factores son considerados antes de indicarle al usuario la ruta a seguir en la solución de su problema.

Metodología de Aseguramiento de Calidad.

Aseguramiento de 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. Esta apertura es necesaria para facilitar la adopción de estas normas en las empresas y no promover el efecto contrario[4].

Ambas técnicas implican el conocimiento de los requerimientos por parte del personal de desarrollo de software. 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" que representa varios pasos:

- Recabar requerimientos y localizar los conflictos existentes entre ellos.
- Recabar descriptores técnicos y clasificarlos.
- Realizar el Análisis Funcional.
- Establecer el Resultado Final Ideal.
- Encontrar soluciones para las zonas de conflicto.

La metodología propuesta no es necesariamente lineal, existen ciertos pasos que pueden realizarse de manera concurrente lo que aumenta la flexibilidad de la metodología. La figura ZZ describe la arquitectura de la herramienta, donde existen una serie de pasos, que se describen a continuación:

Captura de Requerimientos

Captura de requerimientos del cliente, los cuales son capturados y almacenados en el sistema. Si el usuario lo desea puede clasificar a los requerimientos, para ello tendrá que dar de alta en el proyecto una serie de clasificaciones, y después a cada requerimiento asociarlo con alguna de estas clasificaciones.

Búsqueda de conflictos de requerimientos

Una vez capturados y clasificados todos los requerimientos, se procede a analizar las relaciones que existen entre ellos con el fin de buscar si existe algún conflicto, por ejemplo, que un requerimiento contradiga a otro, que un requerimiento afecte a otro, etc. Después de identificar los conflictos hay que eliminarlos proponiendo

[4] El efecto contrario sería el de adaptar las empresas a las normas, algo demasiado restrictivo.

nuevos requerimientos o modificando y eliminando algunos otros. Este punto deja intencionalmente abierta la adopción de una metodología especifica de búsqueda de conflictos para evitar limitar al usuario.

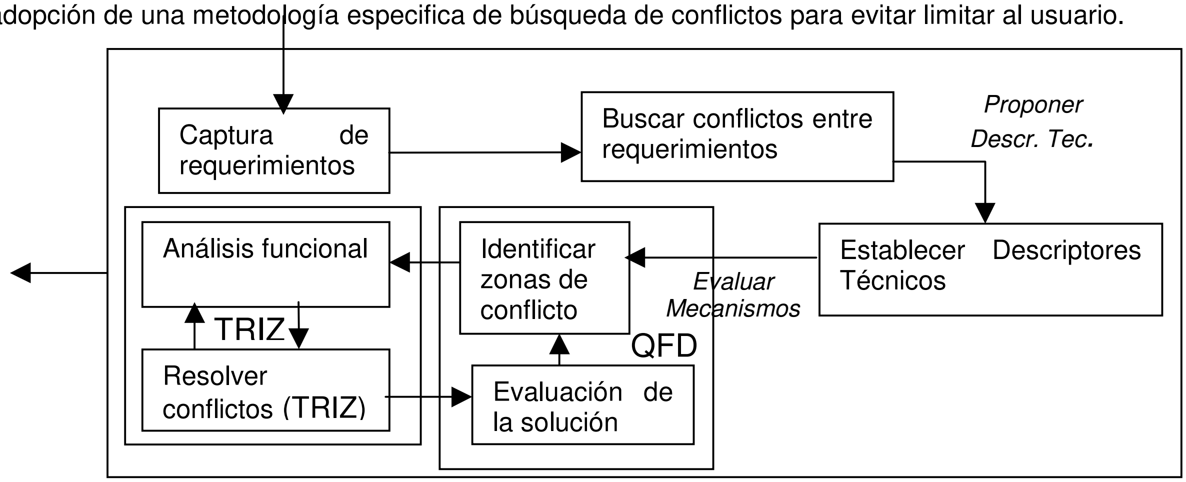


Figura 4 Fases de la Metodología

Establecimiento de los Descriptores Técnicos.

Con los requerimientos depurados se identifican los descriptores técnicos que satisfagan los requerimientos. Es posible clasificar a los DT, indicando un conjunto de categorías, asignando cada DT a la categoría seleccionada.

Identificar Zonas de Conflicto, Análisis Funcional, Resolver Conflictos (TRIZ).

Una vez que se cuenta con todos los DT, se procede a la identificación de conflictos entre ellos, por ejemplo por medio del análisis funcional. Por cada conflicto encontrado, este se clasifica dentro de alguno de los 23 parámetros de contradicción, una vez hecha la clasificación se procede a resolverlos utilizando la metodología de TRIZ.

El mapeo de los DT a parámetros de contradicción permite identificar los principios de inventiva asociados a este conflicto. Cada principio de inventiva tiene asociado una conjunto de soluciones que se han propuesto en algún otro proyecto, si ninguna le satisface entonces se puede proponer una nueva, la cual automáticamente quedara registrada en la base de conocimientos que controla toda las soluciones asociadas a los principios de inventiva.

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 propuesto permite medir la diferencia entre el valor ideal y el porcentaje de avance logrado gracias a la aplicación de la metodología. Esta evaluación permite controlar las iteraciones necesarias para determinar el nivel deseado de calidad.

Para visualizar el avance se evalúa el porcentaje de calidad lograda hasta el momento, él calculo de tal porcentaje de avance que se ha logrado respecto al resultado final ideal, el cual es el valor del proyecto sin pérdidas.

Método de seguimiento de la calidad.

El objetivo de formular el IFR es eliminar el trabajo duplicado (resolver el problema la primera vez) ubicando la raíz del problema o la necesidad del usuario. IFR ayuda a alcanzar avances en soluciones 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, donde idealidad esta definida como:

IA = Suma_de_beneficios / (Suma_de_Costos + Suma_de_perdidas)   (FID)

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 resultado final ideal tiene todos los beneficios, ninguna pérdida y ningún costo extra al presupuestado para el problema original. El sistema ideal no ocupa espacio, no pesa, no requiere labor y no requiere mantenimiento. El IFR tiene 4 características:

1. Elimina deficiencias del sistema original.
2. Preserva las ventajas del sistema original.
3. No hace el sistema mas complicado (usa recursos gratis o disponibles).
4. No introduce nuevas desventajas.

Por lo tanto cuando formulemos un IFR, podemos compararlo con estas cuatro características y la ecuación de Idealidad. 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, cuando no existen conflictos entre los WHATS el valor representa el máximo beneficio obtenible.
La *suma de costos* es el valor obtenido de sumar todos los valores asignados a los HOWS, como se explico en la sección de manejo de requerimientos. Representa el costo subjetivo de llevar a cabo el proyecto, normalmente este valor debe variar muy poco porque un proyecto bien diseñado debe mantener el presupuesto asignado originalmente.

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. Es decir, la suma del total de intersecciones de la pirámide How’s vs How’s, y What’s vs What’s menos las intersecciones que tengan los DT o los requerimientos consigo mismo.
Se define la formula que permite obtener el valor del IFR ideal específico al proyecto, la cual se obtiene eliminando el parámetro “suma de pérdidas asignándole el valor de 0.

IFR = Suma_de_beneficios / (Suma_de_Costos + 0)

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

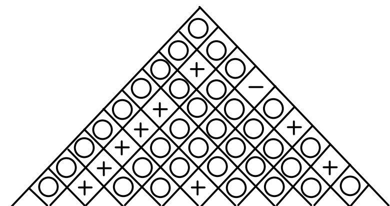
PAC = (IA / IFR) * 100

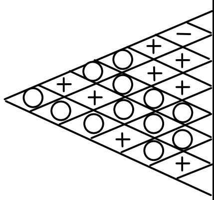
Ejemplo de aplicación:

Evaluación de una herramienta de software para QFD.

La complejidad del desarrollo de software requiere del empleo de herramientas para automatizar los procesos que sean factibles de serlo. En esta sección se presenta la aplicación de la metodología de este artículo para el análisis de una herramienta CASE, por razones de espacio sólo se discuten los aspectos principales de este proyecto y los resultados de la aplicación de la metodología.

Se asume que existe un documento inicial de requerimientos, y que el análisis a desarrollar se enfoca exclusivamente a los aspectos de control y aseguramiento de calidad necesarios en una herramienta CASE. Los valores mostrados pueden considerarse como obtenidos por consenso del equipo de desarrollo, pero pudieron ser obtenidos por cualquier método de círculos de calidad.





	Importancia de los requerimientos	**Descriptores técnicos**	Intercambio de datos via ASCII y con hojas de calculo	Visualización	Ordenamiento de datos acorde a algún criterio definido	Administración de versiones y variantes	Guías de solución	Base de datos de conocimientos	Evaluación del IFR	Seguridad	Análisis de la competencia	Trabajo en grupo	**Competencia**	QFD	QFD extendido	TRIZ	QFD/TRIZ
Dificultad de los descriptores técnicos			0.5	4.5	1.0	2.0	3.7	3.0	1.0	1.0	1.5	3.5					
Requerimientos																	
Clasificación de datos	3.0			5.0	3.0		1.0							0.8	0.0	0.7	0.0
Jerarquización	4.0			5.0	3.0		1.0	1.0						1.0	0.0	1.0	1.0
Identificar y solucionar conflictos	5.0			5.0	1.0	1.0	1.0	5.0	3.0		1.0			0.5	0.5	0.8	0.5
Evaluar la solución	5.0			5.0		1.0		5.0	5.0					0.0	1.0	0.7	0.0
Intercambio de datos	5.0		3.0											0.5	0.0	0.5	0.1
Ergonomía	5.0		1.0	1.0	1.0	1.0	1.0					1.0		0.8	0.1	0.8	0.8
Trabajo con varios usuarios	2.0					3.0		3.0		5.0		5.0		0.0	0.0	0.0	0.0
Importancia de los Descriptores técnicos			20	70	31	21	17	60	40	10	5	15					

1. Requerimientos del sistema.

El primer paso es establecer la lista de requerimientos del sistema, misma 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).

2. Descriptores técnicos.

Una vez con los requerimientos bien definidos, el equipo de desarrollo de software establece los elementos a utilizar para dar solución a dichos requerimientos, es decir se establecen los descriptores técnicos del sistema. De la misma manera que sucedió con los requerimientos 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.

3. 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. Es importante remarcar que en el campo de la investigación, la “competencia” la define el estado del arte del área. Pensar que este método es solo aplicable a negocios puede no ser justo.

4. Identificación de conflictos.

Establecido el listado de descriptores técnicos, la siguiente tarea es la detección de conflictos, aplicando el análisis funcional. Aplicando análisis funcional se detecta un conflicto entre los descriptores técnicos siguientes:

No.	Descriptor técnico	Complejidad
1	Administración de versiones y variantes	2.0
2	Trabajo en grupo	3.5

Con los cuales realizamos el siguiente análisis previo:

Declaración de Función			**Análisis**	**Depurar**	
A	Efectúa esto a	B	Util o Dañino	Es Necesaria la Función	¿Puede B Hacerlo? ¿Como?
1	Controla las versiones múltiples	2	útil	Sí	No

Sin embargo , este análisis no refleja el problema real: ¿Que hacer con el manejo simultáneo de versiones generadas por el trabajo en grupo? Este problema se detecta por la aplicación inversa de FA:

Declaración de Función			**Análisis**	**Depurar**	
A	Efectúa esto a	B	Util o Dañino	Es Necesaria la Función	¿Puede B Hacerlo? ¿Como?
2	Genera versiones simultáneas	1	Dañino	Sí	No

La aplicación del FA permite detectar el problema de la consistencia entre las versiones generadas por actividades simultáneas de trabajo en grupo

5. Primer calculo de seguimiento de la calidad.

A partir de este momento se verifica el producto a través del índice de calidad. El sistema da el valor de idealidad actual, basado en la fórmula:

IA = Suma_de_beneficios / (Suma_de_Costos + Suma_de_pérdidas)

Con los siguientes parámetros:

- Suma de beneficios = 21.5 La suma del total de los requerimientos sin conflictos entre ellos.
- Suma de costos = 21.7 La suma del total de valores de los descriptores técnicos.

- Suma de pérdidas=5.5 Obtenido del conflicto detectado anteriormente.

De donde:

IA= 21.5/(21.7+5.5) = 0.7978
IFR= 21.5/21.7 = 0.9907

Con un porcentaje de avance: PAC = (0.7978/0.9907)*100 = 80.5 %

6. Proyección del conflicto con algún parámetro de contradicción.

El parámetro de contradicción asociado es FIABILIDAD, dado que las versiones que se van guardando probablemente no sean siempre las deseadas.

7. 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 problemas es el principio de "*Equipotencialidad*".

Equipotencialidad ⇒Limitar los cambios

Ahora verificamos las soluciones agrupadas bajo el principio de Equipotencialidad, para nuestro ejemplo suponemos la carencia de soluciones y por tanto se considera un nuevo caso, generando una solución a la medida, expresada como:
¨ Control de la concurrencia de almacenamiento de versiones. Si dos usuarios desean almacenar la versión que están trabajando del proyecto se guarda la primera y se avisa al segundo usuario que existe una versión más actual que la que el desea almacenar. ¨
Finalmente resuelto el conflicto el sistema re-calcula el porcentaje de idealidad:

IA= 21.5/(21.7+0.0) = 0.9907
IFR= 21.5/21.7 = 0.9907

Con un porcentaje de avance: PAC = (0.9907/0.9907)*100 = 100 %
Lo que implica un proyecto sobre el cual es posible iniciar un trabajo con bases firmes.

8. Resultados de QFD.

Finalmente se verifican los datos proporcionados por la matriz de correlación de QFD, es el valor de la importancia técnica. Los valores que indican la importancia atribuida a cada descriptor técnico se enlistan en la parte inferior de la casa de la calidad debajo de su correspondiente descriptor.

Conclusiones y trabajo futuro.

El asegurar la calidad de un producto debe se una actividad mayor en el proceso de desarrollo de SW. La metodología presentada permite obtener un conjunto de requerimientos consistente 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 SW, ya que atacan problemas normalmente no contemplados obteniendo soluciones que permiten aprovechar los conflictos para mejorar el producto. Esta metodología ha sido automatizada en un prototipo lo que permite una mayor flexibilidad de manejo.
Actualmente se trabaja en este prototipo para mejorar la búsqueda de soluciones y para generar una base de casos que ayuden a guiar esta búsqueda. Igualmente se trabaja en la validación de esta metodología por la aplicación en la resolución de casos reales.

Referencias.

- **ANSI/IEEE** Std. 830-1984, IEEE Guide to Software Requirements Specifications.
- **Becker,** The House of Quality. Becker Associates Product Planning, 1997.
- **Dean, E. B.,** Seven Management (New) Tools from the Perspective of Competitive Advantage, 1997.
- **Domb, E.,** The 39 features of Altshuller's contradiction matrix, novembre 1998.
- **Domb, E.,** "The Ideal Final Result: Tutorial", 1997.
- **Gordon G. and G. Shulmeyer,** Zero defects software, Software Engineering series, Mc Graw Hill. 1990 ISBN-N-0-07-055663-6
- **Hales, R. F.,** QFD and the Expanded House of Quality. ProAction Development, Inc, 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, R. and A. Jarvis,** ISO 9000-3, A Tool for Software Product and Process Improvement, Ed. Springer 1995, ISBN-0-387-94568-7.
- **Lamia, W. M.,** Integrating QFD with Object Oriented Software Design Methodologies. Software Engineering Institute, Carnegie Mellon University, 1997.
- **Marconi, J.,** "ARIZ: The algorithm for inventive problem solving", 1984.
- **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, J.,** The QFD, TRIZ and Taguchi connection: customer-driven robust inovation, Ideation International Inc. The Ninth Symposium on Quality Function Deployment, June 1997.
- **Tate, K. and E. Domb,** The 40 inventive principles and the accompanying examples, May 1997.

Anexo A.

Los parámetros de contradicción considerados en el desarrollo de software se describen a continuación.

Parámetros de contradicción	**Comentarios**
Importancia relativa en el sistema (dinámico)	Valor aportado por la creación dinámica de un objeto.
Importancia relativa en el sistema (estático)	Valor aportado por la permanencia estática de un objeto
Parámetro que describe el tamaño del objeto (estático)	Tamaño del objeto. Espacio en memoria.
Parámetro que describe el tamaño del objeto (dinámico)	Tamaño del objeto. Espacio en memoria, liberación de memoria.
Eficiencia	Eficiencia de una función y/o parte del sistema.
Carga de trabajo	Carga de trabajo por procesador.
Interfaz	Definición clara de la interfaz, pocas interfaces, interfaces pequeñas.
Composición del objeto	Composición de un objeto por utilización de otros.
Robustez	Manejo de aserciones. Descripción de precondiciones, postcondiciones, invariantes.
Ineficiencia	Desperdicio de recursos, buscar alternativas de algoritmos, evaluar la adaptación del algoritmo a las estructuras de datos. Sacrificar eficiencia por legibilidad.
Pérdida de información	Falta de control en la información, falta de claves de acceso, malas políticas de respaldo. Fragmentación y replicación como métodos de control de información.
Desperdicio de tiempo	Mala estructura, demasiados objetos actuando como interfaces.
Fiabilidad	Manejo de errores, utilización de objetos redundantes, aplicación de técnicas de tolerancia a fallas.
Precisión de resultados	Cálculo de valores de manera redundante, cambio de algoritmos, uso de representaciones adecuadas (doble precisión, reales,...). Evitar el casting de tipos.
Número de errores	Bitácora de operación, aplicación de auditorías rigurosas, actualización del equipo.
Mal diseño	Evaluar el diseño actual, aplicar un análisis cuidadoso para recuperar lo posible y modificar el diseño.
Mal sistema	Evaluación de la eficiencia, documentación y seguimiento del sistema para evaluar su grado de inadaptación.
Facilidad de programación	Proporcionar las funciones necesarias de manera clara con las opciones para alteración de los parámetros de operación.
Usabilidad	Facilidad de uso. Utilizar prácticas adecuadas de interfaces hombre-máquina, controlar las opciones posibles, evitar encumbrar al usuario con información innecesaria. Mejorar el diseño de las interfaces
Mantenimiento	Control de la operación del sistema, control de errores, documentación, evaluación de los cambios para determinar una mejor versión.
Portabilidad	Capacidad para transportar el sistema a un ambiente distinto de donde fue implementado. Utilización de estándares, lenguajes de uso común.
Complejidad	Evitar diseños complejos, tener claros los objetivos del sistema.

Productividad	Evaluación de la eficiencia para determinar la productividad del sistema. Evaluar la eficiencia con respecto a las mejoras del trabajo global de la organización.

Anexo B.

Principios	Descripción	Comentarios
Modularidad	Dividir un objeto en partes independientes. Hacer un objeto fácil de desensamblar o incrementar el grado de división.	Asegurar que la modularidad es la correcta. Los módulos deben corresponder a unidades lógicas del sistema en cuestión.
Extracción	Considerar la remoción o aislamiento de la parte o la propiedad o acción que causa la interacción indeseable.	Asegurar el grado adecuado de encapsulamiento. Lo que permite el aislamiento de relaciones o propiedades no deseables.
Calidad Local	Cambiar los parámetros de funcionamiento ya sea externos o internos.	Verificar los algoritmos que se emplean en cada objeto. En ocasiones es aceptable el sacrificio de la eficiencia por la legibilidad.
Asimetría	Cambiar la forma del objeto de simétrico a asimétrico. Privilegiar una función sobre las otras.	Especialización de funciones privilegiadas. Investigar las posibilidades de emplear polimorfismo.
Combinación	Juntar o combinar objetos similares o idénticos o ensamblar partes que ejecutan operaciones paralelas.	Aplicar la Herencia múltiple al diseño de objetos.
Universalidad	Genericidad.	Concebir el sistema por medio de clases abstractas que permitan controlar el grado de refinamiento.
Anidamiento	Poner un objeto dentro de otro.	Composición de objetos. Manejar correctamente la agregación, la visibilidad y el tiempo de vida.
Contrapeso	Compensar el peso de un objeto con otros que carguen o balanceen ese peso.	Prioridades de ejecución, reordenamiento de acciones. Balance de carga.
Contra-acción preliminar	Si es necesario, ejecutar una acción con ambos efectos dañinos y útiles.	Revisar el diseño. Sólo en caso estrictamente necesario considerar esta posibilidad.
Acción preliminar	Acción de acondicionamiento previo.	Equivalente al principio de divide y vencerás. Implementar objetos filtros que ejecuten una acción previa antes de cargar al objeto final con trabajo innecesario.
Compensación	Prevenir acciones de amortiguamiento de fallas en caso de objetos poco confiables.	Considerar el uso de aserciones dentro del diseño. Emplear las cláusulas TRY y CATCH normalmente encontradas en JAVA y C++
Equipotencialidad	Limitar el cambio de posición.	Dar la misma importancia a ciertas acciones para controlar su aparición. Control de concurrencia.
Reversa	Invertir las acciones utilizadas para resolver un problema. Por ejemplo hacer móviles partes fijas.	Alterar el orden de solución del problema. Buscar alternativas.
Grado de dinamismo	Permitir que las características de un objeto o proceso cambien para ser óptimas, encontrar la condición óptima de operación.	Alterar el tipo de control de ejecución. Por ejemplo usar semáforos, monitores, para asegurar el orden de ejecución.

Exceso o carencia	Usar un poco menos o un poco más de un ingrediente.	Si un problema se resuelve con un objeto, tratar con dos o viceversa.
Cambio de dimensión	Alterar las dimensiones del objeto.	Cuestionar las clases abstractas, los módulos, los subsistemas.
Acciones periódicas	En lugar de acciones continuas usar acciones periódicas.	Cambiar el enfoque de procedural a un manejo por eventos.
Acciones útiles estables.	Mantener las acciones útiles trabajando de manera continua.	Tratar de adelantar el trabajo, mantener las funciones útiles disponibles en memoria.
Correr a través (rushing Through)	Aumentar la velocidad del proceso de ciertas etapas.	Tener funciones más sencillas de algoritmos. Sacrificar exactitud por eficiencia.
Cambiar un menos en un más	Cambiar limones en limonada. Usar los efectos peligrosos para conseguir el efecto deseado.	Aprovechar los resultados no utilizables directamente, aprovechar la presencia de un valor peligroso para inferir la presencia de errores
Realimentación	Introducir realimentación para mejorar un proceso o acción	Medir las salidas del sistema para detectar posibles desviaciones y corregirlas.
Intermediación	Utilice un elemento o proceso intermediario.	Utilización de objetos filtros que agrupen el chequeo de los valores utilizables en funciones.
Autoservicio	Haga que el elemento se auto proporcione los elementos que requiere para funcionar.	Idem.
Copiado	En lugar de utilizar elementos caros o poco disponibles utilice copias baratas del mismo elemento.	Utilizar la tecnología de agentes. Utilizando el modelo del pizarrón. Buscar la solución por n-agentes en lugar de programar el problema de un sólo golpe.
Vidas cortas baratas	Reemplace un objeto complejo con copias múltiples de objetos baratos cuyo conjunto comprende las cualidades del objeto original (p.ej. vida de servicio), p. Ej. En lugar de un multiplicador de 32 bits utilice dos de 16.	Utilizar la tecnología de agentes, manejando el trabajo por fases. Aplicar diferentes pizarrones para la solución de problemas,
Rediseño	Reconsidere la alteración total del objeto u objetos involucrados. Cambie de campos estáticos a dinámicos, de campos sin estructura a campos estructurados, use campos en conjunción con partículas activadas por campos.	Reconsiderar la alteración total de la estructura de diseño. Altere el foco de control de cada operación, intente alterar el grado de concurrencia. Intente cambiar el estilo de procedural a manejado por eventos.
Membranas flexibles	Use capas y membranas flexibles y delgadas en lugar de estructuras tridimensionales. Aislar el objeto del ambiente externo usando capas flexibles.	Aumentar la visibilidad del objeto, ya se en número de funciones o en permisos de acceso. Restringir el acceso. Verificar la modularidad de cada objeto. Verificar que las clases abstractas capturen la comunalidad de manera correcta.
Materiales porosos	Haga poroso un objeto o agregue materiales porosos. Si ya es poroso introduzca funciones útiles.	
Homogeneidad	Haga que los objetos interactuen con objetos de propiedades idénticas. Ejemplo corte un diamante con un diamante	Separe la implementación de la política. Es posible que el problema mayor se divida en 100 problemas menores equivalentes manejados por 100 objetos iguales.
Rechazo y	Haga dinámicas porciones de un objeto	Los recursos necesarios varían en

regeneración	por ejemplo si una parte ya desarrollo su función debe desaparecer y regenerarse solo cuando sea necesario. Eventualmente usando refacciones.	condiciones normales y en condiciones críticas. Es posible generar agentes computacionales para tratar con situaciones específicas.
Cambios de propiedades	Cambio del estado físico de un objeto.	Analizar si el objeto no puede ser expresado de acuerdo a sus estados, en lugar de sus procedimientos.
Uso de transiciones de fase	Utilice los fenómenos que ocurren durante la transición de fases. Agua cambiando a vapor.	Verificar que acciones pueden ejecutarse debido a la ocurrencia de un evento.
Ambiente inerte	Reemplace el ambiente normal con uno inerte. Anexar partes neutras o aditivos inertes a un objeto.	El uso de un buffer permite desacoplar las dinámicas de objetos diferentes. Investigue el uso de archivos como elemento de desacoplo. Verificar si no existen memorias compartidas.
Materiales compuestos	Cambiar de materiales uniformes a compuestos.	Utilizar elementos de otros desarrollos, por ejemplo interoperabilidad con bases de datos. Utilización de rutinas escritas en lenguajes diferentes. Interoperabilidad con otros sistemas.