CENTRO DE INVESTIGACION EN MATEMÁTICAS, A.C.

Clasificación y Análisis Post-Proceso de Defectos en Proyectos de Desarrollo de Software

Reporte Técnico de Investigación
que para obtener el grado de

Maestro en Ingeniería de Software

presenta

Gustavo Izcóatl Herrera Romero

Director:

Dr. Carlos Montes de Oca

Guanajuato, Gto. a 22 de julio de 2005.

Clasificación y Análisis Post-Proceso de Defectos en Proyectos de Desarrollo de Software

Versión 1.0
Julio de 2005

Histórico de revisiones

Versión	Fecha	Descripción	Autor
0.1	21-07-2005	PROPUESTA INICIAL	GIHR
1.0	31-07-2005	VERSIÓN FINAL	GIHR

AGRADECIMIENTOS

Quiero agradecer a mis profesores durante la maestría: los doctores Carlos Montes de Oca, Cuauhtémoc Lemus Olalde, Miguel Serrano Vargas, y el Maestro Gerardo Padilla, por transmitirnos sus conocimientos e ideas, así como su amplia experiencia respecto al desarrollo de software.

Agradezco a mis compañeros de maestría, así como a mis muchos amigos en el CIMAT, por su amistad, y por su apoyo.

Este trabajo fue parcialmente financiado por el Consejo de Ciencia y Tecnología del Estado de Guanajuato (CONCYTEG) a través del proyecto GTO-2002-C01-5333 titulado "Promoviendo calidad en la industria de software: recursos humanos, investigación, servicios".

Tabla de Contenido

FIGURAS

TABLAS

1 Introducción

Si uno desea un sistema de software de alta calidad, uno debe asegurarse de que cada una de sus partes sea de alta calidad. Al mejorar la administración de los defectos, uno producirá componentes de software más confiables. Por lo tanto, para mejorar la calidad del producto, debe mejorarse la calidad del proceso. Esto requiere medir y tener un seguimiento de la calidad del proceso, además del producto.

La definición de calidad tiene como punto principal las necesidades y perspectivas del usuario. Se define la calidad como cumplimiento de los requisitos o necesidades del cliente. Esto es natural, ya que si un producto de software no realiza las tareas en la forma que un usuario necesita, en el momento en que las necesita, no juzgará al producto como de alta calidad. Sin embargo, aunque todo mundo acepta lo antes dicho, en lugar de usar los recursos del proceso de desarrollo para hacer que el producto cumpla con las expectativas del cliente, en realidad los dedican a realizar pruebas. Y el costo de las pruebas se dedica casi exclusivamente a encontrar y reparar defectos.

Los costos de encontrar y corregir un defecto incluyen el costo de una o varias de estas actividades:

- Determinar que hay un problema.
- Encontrar la causa del problema.
- Determinar qué está mal en el producto.
- Reparar los requerimientos.
- Reparar el diseño.
- Reparar la implementación.
- Revisar la reparación para asegurarse de que es correcta.
- Probar la reparación para asegurarse de que resuelve el problema identificado.
- Probar la reparación para asegurarse de que no provoca otros problemas.
- Cambiar la documentación donde sea necesario para reflejar la corrección.

De esto se deduce que mientras más tiempo permanezca un defecto en el producto, más caro será corregirlo al encontrarlo, y más elementos se verán involucrados en la corrección. Por ejemplo, encontrar un defecto de requerimientos en la etapa de pruebas puede ser muy caro, pero encontrar un error de codificación en la inspección de código generalmente costará mucho menos. El objetivo es entonces encontrar los defectos tan pronto como sea posible, con el fin de minimizar el número de defectos en el producto en cada etapa, y minimizar la cantidad de trabajo de reparación, y el costo que esto trae consigo.

2 Definición del problema

2.1 Descripción

Se tiene un conjunto de defectos registrados en una tabla. Para cada defecto se cuenta con las siguientes propiedades o datos:

- Clave, nombre y tipo de proyecto.
- Clave, nombre y apellido del desarrollador que cometió el defecto.
- Clave del tipo de componente.
- Descripción del tipo de componente
- Nombre del componente.
- Fase de inserción.
- Fase de eliminación.
- Clave y descripción del tipo de defecto.
- Clasificación de la importancia del defecto.
- Descripción del defecto.

El problema consiste en:

- Encontrar una manera de clasificar este conjunto de defectos.
- Analizar la clasificación hecha para obtener métricas de calidad del proceso tales como densidad de defectos, y efectividad de detección y remoción de defectos.
- Observar en estas métricas, posibles debilidades en el proceso de detección y remoción de defectos.
- Generar recomendaciones para mejorar el proceso de desarrollo de software actual, con el fin de mejorar la detección y remoción de defectos, así como su prevención futura.

2.2 Justificación.

La clasificación y análisis de este conjunto de defectos es importante, ya que proveerá una retroalimentación acerca de las partes del proceso de desarrollo de software donde se puede hacer un mejor trabajo, traducido esto en una mejora del proceso que evita la recurrencia de los defectos más comunes y los más costosos, ahorrando tiempo de reparaciones y aumentando la productividad del equipo de desarrollo.

Al conocer el comportamiento del proceso de desarrollo respecto a los defectos cometidos y la efectividad de las actividades de detección y remoción, se identifica las mejoras al proceso de manera que mejore la efectividad de de la detección y la remoción, logrando encontrar los errores más temprano en el proceso de desarrollo, disminuyendo con esto de manera importante los costos de reparación. Asimismo, al conocer los defectos más comunes o los más costosos que se cometen con regularidad, es posible mejorar el proceso de desarrollo para evitar cometer de nuevo esos defectos (prevención de los defectos),

aumentando además de la calidad del producto, la productividad del equipo de desarrollo, al dedicar más tiempo a las actividades reales de creación, y menos a las de corrección.

2.3 Metodología de investigación.

La metodología seguida para llevar a cabo la solución del problema descrito fue:

- Búsqueda de información sobre los métodos conocidos de clasificación y análisis de defectos.
- Lectura y análisis de la información recabada.
- Propuesta de un procedimiento de clasificación y análisis de los defectos.
- Análisis de los datos utilizando el procedimiento propuesto.
- Análisis de los resultados del procedimiento propuesto, para generar recomendaciones específicas y generales al proceso de desarrollo actual.

3 Estado del arte

A continuación se describirá algunos de los trabajos más importantes por su efectividad en el área de clasificación y análisis de los defectos encontrados en sistemas de software, así como del análisis de la efectividad de remoción de defectos de un proceso de software. Las metodologías que a continuación se describen se distinguen por analizar los defectos encontrados en un sistema, tanto durante el proceso de desarrollo como al final de este, y encontrar posibles debilidades y mejoras en el proceso, con el fin de evitar la recurrencia de los defectos, y mejorar la efectividad de la detección y remoción de las actividades dedicadas a la mejora de la calidad del producto.

3.1 Clasificación ortogonal de defectos.

Se trata de un esquema desarrollado por IBM para clasificar defectos con el fin de proveer retroalimentación sobre el progreso de un proyecto, a través del mismo. El objetivo de su creación es obtener un paradigma para medir las relaciones causa-efecto de los defectos, donde se pudiera encontrar la causa raíz de un defecto. Al mismo tiempo se deseaba desarrollar una clasificación de defectos que fuera simple y que estuviera lo más libre posible de errores humanos.

A partir de las investigaciones que permitieran obtener esta clasificación se demostró que al clasificar de forma única los defectos utilizando un conjunto de tipos de defectos que reflejaran la semántica de las reparaciones hechas, se puede asociar los tipos de defectos con las fases de desarrollo.

ODC significa esencialmente que se clasifica a un defecto en uno de varios posibles tipos, los cuales apuntan a la parte del proceso que necesita atención.

Para esto se agregó dos atributos a los defectos: tipo de defecto y desencadenador (*trigger*) de un defecto.

El tipo de defecto clasifica a un éste a través de la semántica de la reparación que se necesitó hacer para remover al defecto del sistema. La selección del tipo de defecto se deriva de la corrección hecha por el desarrollador. Para cada uno de estos tipos, el desarrollador también indica si el defecto consiste en que la información en el componente hace falta, o si es incorrecta.

Los tipos de defectos definidos son:

- Función. Afecta las interfaces de usuario, las interfaces del producto, las interfaces con el hardware, y las estructuras de datos. Requiere un cambio formal en el diseño.
- Asignación. Generalmente indica errores en el código, al inicializar o asignar valores de variables o estructuras de datos.

- Interfase. Se refiere a errores en las interacciones entre componentes, módulos u otros dispositivos a través de las llamadas a funciones, paso de parámetros, etc.
- Chequeo. Errores en la lógica del programa, el cual no hace la verificación de los datos y valores antes de ser usados.
- Serialización/temporal. Errores que tienen que ver con los recursos compartidos y de tiempo real.
- Construcción/Empaquetado (*Build/Package/Merge*). Errores que ocurren debido a problemas en las librerías, así como problemas en el control de cambios y versiones.
- Documentación. Errores que pueden afectar a la documentación de los artefactos o al mantenimiento del sistema.
- Algoritmo. Problemas de eficiencia o corrección de una tarea, a través del algoritmo que define a esa tarea.

Los tipos de defectos deben ser ortogonales, es decir, que son mutuamente excluyentes. Esto asegura que al encontrarse un defecto, este será solamente de un tipo posible, y no podrá caerse en la situación de clasificar incorrectamente a un defecto, ya que no será posible clasificar a un defecto en dos posibles tipos. Al mismo tiempo, los tipos de defectos definidos deben cubrir todas las posibles ocurrencias de los defectos en un sistema, de manera que ningún defecto pueda quedar sin clasificación.

El objetivo de los tipos de defectos también es poder relacionar cada tipo con algunas de las fases del proceso de desarrollo. Se trata de determinar dónde fueron inyectados los defectos en el sistema. Se encontró que cada tipo de defecto tiende a ser inyectado en relativamente pocas áreas o fases de desarrollo. Por ejemplo, los defectos de asignación tienden a ser inyectados en su mayoría en la fase de codificación, y los defectos de algoritmo tienden a ser inyectados especialmente en la fase de diseño de bajo nivel.

La asociación encontrada entre los tipos de defectos y el proceso de desarrollo se muestra en la siguiente tabla:

Tipo de defecto	**Asociación al proceso.**
Función	Diseño
Interfase	Diseño de bajo nivel
Chequeo	Diseño de bajo nivel o código
Asignación	Código
Serialización y temporales	Diseño de bajo nivel
Construcción/Empaquetado	Librerías
Documentación	Documentación
Algoritmo	Diseño de bajo nivel

Tabla 3.1 Relación entre tipos de defectos y fases de desarrollo.

Al final de cada fase de desarrollo, se generan gráficas que muestran el número de defectos de cada tipo encontrados en esa fase. Al comparar el número de ocurrencias de un tipo de defecto por fase, el cambio en la distribución da una retroalimentación sobre el estado del

proceso de desarrollo. Por ejemplo, si las fases de desarrollo fueran diseño, codificación, pruebas de unidad, pruebas de integración y pruebas de sistema, se puede esperar encontrar un mayor número de defectos de tipo algoritmo en la fase de codificación y de pruebas de unidad (poco después del diseño de bajo nivel, que es cuando este tipo de errores se inyecta en mayor número), y un número cada vez menor en las siguientes etapas. Si la curva de este tipo de defectos no es como se espera, (por ejemplo, si se sigue encontrando un alto número de defectos de tipo algoritmo en las pruebas de integración o de sistema) posiblemente se ha salido de la fase de codificación o de la fase de pruebas de unidad de forma prematura.

Dado que las distribuciones de los tipos de defectos cambian con el tiempo, proveen una forma de medir la madurez del producto. Cuando una desviación del proceso se identifica a través de una desviación de las curvas de distribución de defectos, el tipo de defecto encontrado apunta a la parte del proceso que necesita atención.

Al usar una clasificación por tipo de defecto se pueden obtener conclusiones como la anteriormente expuesta, para analizar el proceso de desarrollo. También se puede analizar datos de varios proyectos para encontrar tendencias sobre un ambiente específico de desarrollo (tipo de proyecto, tipo de plataforma, etc.). Posteriormente también es posible realizar análisis numéricos y/o estadísticos, para no solamente encontrar las tendencias, sino definir el desempeño deseado de un proceso de desarrollo, y calibrar el proceso de desarrollo futuro para lograr dichos resultados.

El *trigger* de un defecto es la condición que conduce a una falla, la cual expone al defecto. Al igual que con el tipo de defecto, el conjunto de triggers es ortogonal y comprenden a todo el conjunto posible de situaciones que conducen a fallas. Mientras el tipo de defecto da retroalimentación sobre el proceso de desarrollo, el *trigger* da retroalimentación sobre los procesos de pruebas y verificación. Si la distribución de los *triggers* de los defectos encontrados después de la liberación de un producto (defectos encontrados por el cliente), son distintos de la distribución de los *triggers* durante las etapas de pruebas, posiblemente el proceso de pruebas no está realizando los tipos correctos de pruebas.

Los investigadores de IBM aseguran que ciertos defectos tienden a ser encontrados por inspectores con cierta experiencia específica. Por ejemplo cualquier inspector en una inspección de código debería ser capaz de encontrar defectos cuyo *trigger* sea una falta de cumplimiento del diseño. Sólo inspectores muy experimentados serían capaces de encontrar defectos cuyo *trigger* sea una situación muy rara. De esta manera se tiene una tabla que relaciona los posibles *triggers* y los perfiles de los inspectores que serían capaces de encontrar defectos con esos *triggers*. De esta manera se puede asignar los inspectores necesarios a una tarea de inspección, de forma óptima. La siguiente tabla muestra la relación entre *triggers* de defectos y la experiencia necesaria:

Trigger	Nuevo	Producto	Entre productos	Mucha experiencia
Cumplimiento del diseño	x	x	x	x
• Documento de requerimientos				
• Documento de especificación del diseño				
Entender detalles		x	x	x
• Semántica operacional				
• Efectos colaterales				
• Concurrencia				
Compatibilidad con versiones anteriores		x		
Compatibilidad con otros productos			x	
Situación rara				x
Consistencia y completitud de los documentos		x	x	x
Dependencias de lenguaje	x	x		x

Tabla 3.2 Relación entre tipos de triggers y experiencia necesaria.

Al igual que con los tipos de defectos, se espera que los tipos de *triggers* encontrados por cada etapa de revisión o inspección tengan una cierta distribución. Por lo tanto, las gráficas de distribución de *triggers* por fase de revisión o inspección pueden ser estudiadas para encontrar posibles debilidades en los procesos de revisión, inspección o pruebas que se llevan a cabo en el proceso de desarrollo. Por ejemplo, si la cantidad esperada de un cierto tipo de *trigger* no es encontrada durante una revisión, posiblemente el equipo de inspección no tiene las habilidades adecuadas.

En conclusión, la clasificación ortogonal de defectos de IBM utiliza datos fácilmente obtenidos para realizar inferencias sobre el proceso de desarrollo. Al utilizar datos cualitativos que no dependen de opiniones personales, y tipos y triggers de defectos que son ortogonales, su obtención y análisis son relativamente sencillos.

3.2 Proceso de Prevención de Defectos.

El objetivo de este proceso es, como su nombre lo indica, eliminar la recurrencia de los defectos. Para esto, se analizan los defectos para entenderlos, con el objetivo de no permitir la entrada de defectos similares en el producto en el futuro.

La piedra angular de este proceso son los análisis causales. En éstos cada desarrollador aporta datos objetivos sobre cada defecto. Estos datos son registrados en una base de datos, y al final de cada fase, se realiza una junta para analizar los datos colectivamente para determinar sus causas. Es una junta de dos horas al final de cada fase del proceso de desarrollo, en la cual el equipo de desarrollo discute los defectos registrados en la base de datos. Se lleva a cabo una abstracción del defecto, relacionándolo con un error. Cuando los errores de los defectos han sido abstraídos, son clasificados en estas cuatro categorías:

- Incompleto (Oversight). El desarrollador no consideró completamente algún detalle del problema. Por ejemplo, no se tomó en cuenta un caso posible del valor de una variable.
- Educación. El desarrollador no entendió un proceso o una actividad, debido a una falta de entrenamiento.
- Comunicación. No se recibió la información necesaria, o la información fue transmitida de forma incorrecta, y por lo tanto malinterpretada o no entendida.
- Trascripción. El desarrollador cometió un error tipográfico o de tipo similar. El desarrollador comprendió completamente el proceso o actividad, pero cometió un error.

Los participantes de las juntas discuten los errores entre ellos y tratan de determinar la causa del error y cuándo fue cometido, con el fin de prevenir errores similares en el futuro. Parte del proceso incluye buscar los errores similares aún no encontrados en el artefacto o artefactos analizados. Esto tiene como objetivo la "extinción de los defectos", es decir, remover todas las instancias de ese error y ajustar el proceso de desarrollo para que no vuelvan a ocurrir más instancias de este error.

La última media hora de la junta se dedica a examinar la información de los errores para identificar tendencias y examinar el proceso de desarrollo de la fase previa. Se anota las partes de la fase previa de desarrollo que tuvieron resultados positivos de forma que pueda hacerse sugerencias a otros equipos. Las partes con resultados negativos se identifican, y se anotan las posibles sugerencias de mejora. Al final de la junta, todas las sugerencias de la fase anterior se registran en la base de datos de acciones (*action database*), junto con las causas de los errores descubiertos en la fase actual.

El siguiente paso del proceso involucra al equipo de acción (*action team*). El tamaño de este equipo se calcula aproximadamente como 1 persona por cada 10 a 20 miembros del personal de desarrollo. A este equipo lo conforman miembros del personal de desarrollo de tiempo completo que dedican parte de su tiempo a estar en el *action team*. Este equipo lleva a cabo regularmente juntas en las cuales evalúan todas las sugerencias a implementar, cómo implementarlas, y quién se encargará de dirigir y realizar la implementación.

La retroalimentación es muy importante en este proceso. Los desarrolladores necesitan estar actualizados en los cambios al proceso, así como en los defectos descubiertos para evitar cometerlos. Para proveer esta información, se llevan a cabo al principio de cada fase de desarrollo juntas de comienzo de fase para el equipo de desarrollo. En estas juntas se discute el proceso a seguir durante la fase, de forma que cada desarrollador conozca cómo realizar las tareas de esa fase. Se discuten los resultados deseados para la fase, así como una lista de errores comunes. Esta lista de errores es una compilación de errores que comúnmente suceden en esa fase de desarrollo.

El análisis de aplicar este proceso de prevención de defectos muestran una reducción de errores por KLOC en aproximadamente 50% comparados con procesos de desarrollo que

no aplican la prevención de defectos. Se estima el costo del esfuerzo de prevención de defectos, incluyendo el tiempo de las juntas y la implementación de las acciones sugeridas, en aproximadamente 0.5% del esfuerzo total de desarrollo.

Además de los datos de defectos registrados durante el proceso de desarrollo, es posible analizar los reportes de defectos suministrados por el cliente. Estos datos pueden ser registrados en una base de datos, y analizados para encontrar posibles problemas en los proceso de revisiones, inspecciones o pruebas.

3.3 Modelo de Remoción de Defectos.

El proceso de remoción de defectos (PRD) utilizado en un proceso de desarrollo, juega un papel crítico para obtener software de alta calidad. La efectividad del PRD depende más que nada en las técnicas de revisión, inspección, verificación y validación utilizadas. Un proceso de remoción de defectos consiste típicamente de fases de remoción de defectos, o subprocesos que involucran pruebas e inspecciones.

Un proceso de desarrollo de software típicamente incluye las siguientes fases:

- Fases de construcción: análisis de requerimientos, diseño, codificación.
- Fases de pruebas: pruebas de unidad, de integración y de sistema.

Las fases de construcción incluyen actividades de remoción de defectos tales como revisiones e inspecciones, las cuales usualmente se llevan a cabo al final de la fase. La función de las fases de pruebas es solamente remoción de defectos.

Las etapas de pruebas por sí solas son inadecuadas para producir software de alta calidad. Las inspecciones proveen una forma efectiva de aumentar la calidad y la productividad. Las inspecciones, según los estudios hechos, pueden eliminar aproximadamente 75 por ciento de todos los defectos, reduciendo al mismo tiempo los costos totales de desarrollo hasta en un 25 por ciento.

El modelo de remoción de defectos (MRD) se ve afectado por la efectividad de las inspecciones y de las pruebas de un proceso de desarrollo. El MRD tiene dos principales aplicaciones. Primera, establecer una relación entre las diferentes fases de remoción de defectos. Y segunda, ayudar a encontrar formas de mejorar el proceso de remoción de defectos.

3.3.1 Datos sobre remoción de defectos.

Los datos de calidad definidos para un producto de software son:

- Densidad de defectos. Es una medida de los defectos presentes en el producto, normalizados por su tamaño. Comúnmente se definen como el número de defectos por cada mil líneas de código (defect/KLOC) o el número de defectos por punto de función (defect/PF).
- Tasa de detección de defectos (T). Es el porcentaje de los defectos totales, detectado en una fase.
- Efectividad de remoción de defectos. También conocido como tasa de remoción de defectos (R) de una fase de desarrollo se refiere al porcentaje de todos los defectos removidos por las revisiones, inspecciones o pruebas hechas durante la fase. A valores mayores de R, mejor la capacidad del proceso de desarrollo para remover los defectos.

La efectividad de remoción de defectos se calcula de la siguiente manera:

$$R = \frac{densidad\ de\ defectos\ durante\ desarrollo - densidad\ de\ defectos\ en\ liberación}{densidad\ de\ defectos\ durante\ desarrollo}$$

De los estudios hechos por Fagan, Rusell y Gilb, entre otros, se obtiene el mejor desempeño de remoción de defectos cuando se incluye en los proyectos las inspecciones formales de diseño, código y otros documentos como requerimientos y manuales de usuario. Los reportes de Fagan indican que las inspecciones pueden encontrar del 60 al 90 por ciento de los defectos, antes de las pruebas de unidad. Rusell reporta que las inspecciones son de dos a cuatro veces más eficientes que las pruebas.

3.3.2 Modelo de remoción de defectos

El proceso de remoción de defectos se modela tomando en cuenta la interacción de la inyección, detección y remoción de los defectos. Esto se muestra en la siguiente figura:

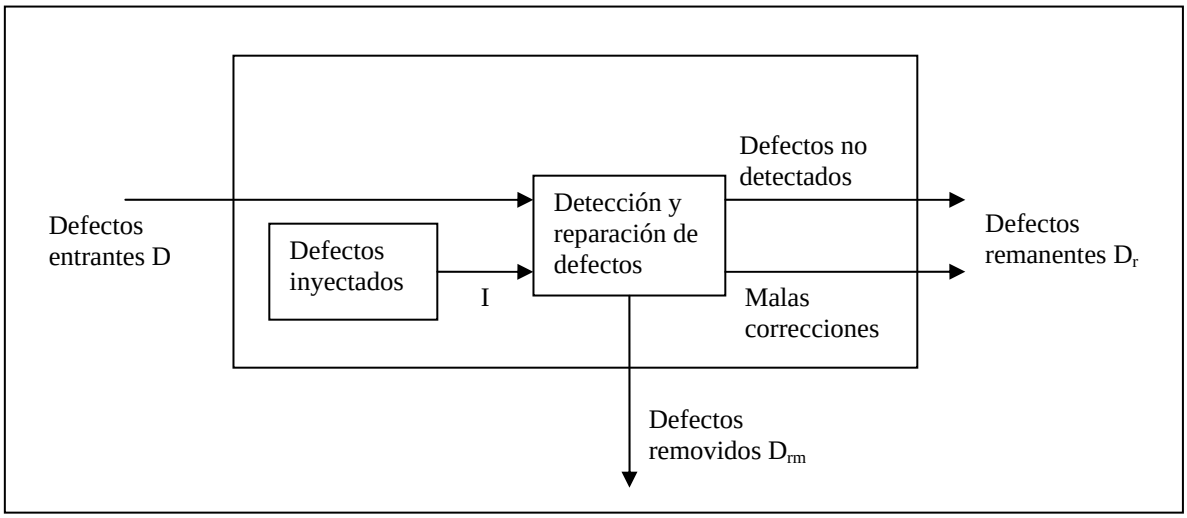

Fig. 3.1 Modelo de Remoción de Defectos.

La inyección de defectos no necesariamente ocurre en todas las fases. En las etapas de construcción, nuevos defectos pueden ser inyectados. Las inspecciones o revisiones se llevan a cabo típicamente al final de las fases de construcción para remover los defectos. Durante las etapas de pruebas, generalmente no se inyectan nuevos defectos. Sin embargo malas correcciones pueden ocurrir.

Sea D los defectos que se entran al proceso de desarrollo, I los defectos inyectados durante el proceso, D_{rm} los defectos removidos, y E la tasa de detección y reparación de defectos. E se define como $D_{rm}/(D+I)$ e incluye el efecto de las malas correcciones. Los defectos remanentes se calculan como $D_r = (D+I)(1-E)$.

Se puede simplificar el modelo, escondiendo los detalles de los nuevos defectos inyectados. Esto nos da el modelo siguiente:

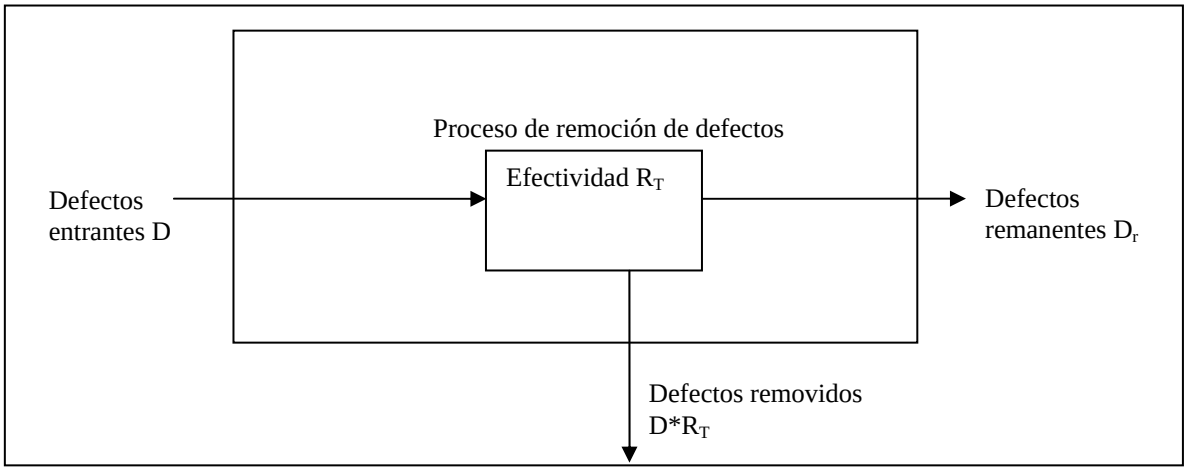

Fig. 3.2 Modelo de Remoción de Defectos simplificado.

Los defectos remanentes después del proceso de remoción de defectos se calcula como:

$D_r = D * (1 - R_T)$

La efectividad de remoción de defectos R_T incorpora el efecto de la detección y reparación de defectos, y la inyección de nuevos defectos.

A continuación se considerará la relación entre los procesos de remoción de defectos individuales que conforman al proceso de remoción general. Para un proceso de desarrollo con dos procesos de remoción de defectos, la relación de remoción se muestra en la siguiente figura:

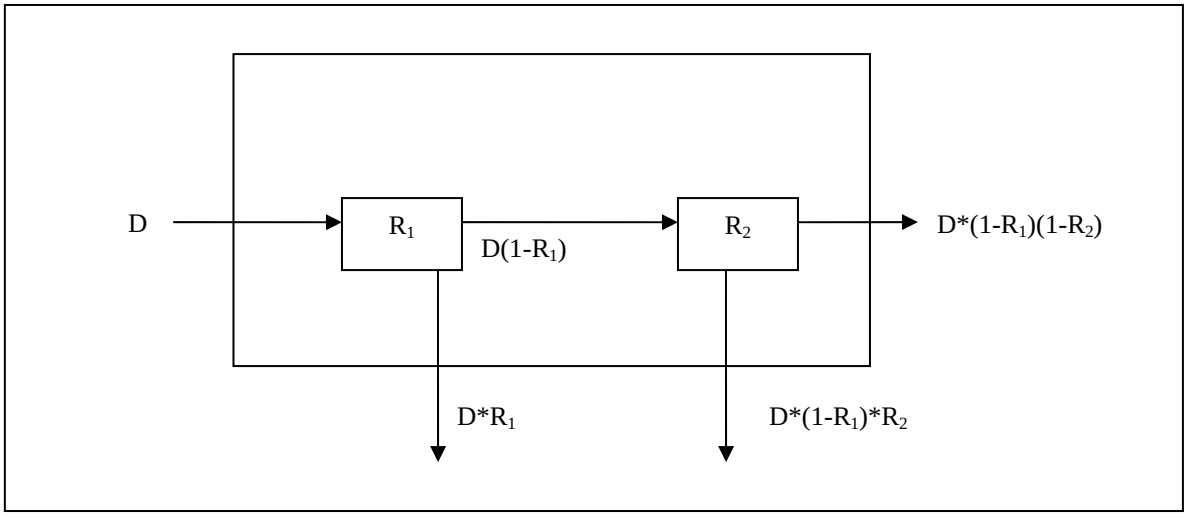

Fig. 3.3 Modelo de Remoción de Defectos con dos fases.

La RT del proceso de desarrollo se relaciona con la efectividad de remoción de defectos de las dos fases de la siguiente manera:
$D*(1-R_{[1,2]}) = D*(1-R_1)*(1-R_2)$ o
$R_T = R_{[1,2]} = R_1 + R_2 – R_1*R_2$

Un análisis similar puede aplicarse al proceso de desarrollo que usa tres fases de remoción de defectos:

$D*(1-R_{[1,2,3]}) = D*(1-R_1) *(1-R_2) *(1-R_3)$ o
$R_T = R_{[1,2,3]} = R_1 + R_2 + R_3 – R_1*R_2 – R_1*R_3 – R_2*R_3 + R_1*R_2*R_3$

3.3.2.1 Mejorando la remoción de defectos

Hay dos maneras de mejorar la remoción de defectos:

a) Agregando una o más fases de remoción de defectos.

b) Mejorando una o más de las fases de remoción de defectos existentes.

a) Agregar una fase de remoción de defectos.

Sea P el proceso actual de remoción de defectos, y P' el nuevo proceso de remoción de defectos con una fase extra de remoción de defectos F_n. Sea R_T la efectividad de remoción de P, R_n la efectividad de remoción de F_n, y sea R' la efectividad de remoción de defectos del nuevo proceso. Para lograr una mejor remoción de defectos, los defectos remanentes de P' no deben ser más que los del nuevo proceso de remoción de defectos. Esto es:

$D*(1-R_T)*(1-R_n) \leq D*(1-R')$

lo cual se puede simplificar como:

$$R_n \geq \frac{R'-R_T}{1-R_T}$$

De esta manera, de forma cuantitativa puede calcularse la efectividad de una nueva fase de remoción de defectos, en función de la efectividad de remoción general deseada.

b) Mejorar una fase de remoción de defectos.

Si se desea mejorar una fase de remoción de defectos, el objetivo sería determinar la mejora requerida R_i de esa fase. Considerando el caso en el que se desea mejorar la primera fase de un proceso de remoción de defectos con k fases. Recordemos que:

$D*(1-R_1)*\ldots*(1-R_k) = D*(1-R_T)$

Si la fase mejorada tiene una efectividad de remoción de R_n y eso provoca una efectividad R' de todo el proceso de remoción, se tiene que:

$D*(1-R_n)*\ldots*(1-R_k) = D*(1-R')$

Combinando las dos ecuaciones anteriores y simplificando:

$$R_n = \frac{(1-R')*(1-R_1)}{(1-R_T)} \qquad i = 1,\ldots k$$

Para una organización que use un proceso de remoción con una $R_T = 0.9$ y una fase $R_i = 0.68$ que desea mejorar, la mejora en esa fase R_i sería de acuerdo a la siguiente tabla:

Tipos de DRP	R' = 0.96	R' = 0.99
2 fases	0.87 (28%)	0.97 (43%)
3 fases	0.83 (57%)	0.95 (79%)
4 fases	0.77 (75%)	0.94 (114%)

Los valores entre paréntesis representan el porcentaje de mejora relativa a la fase R_i. Por ejemplo, si se tiene un DRP de 3 fases, y se desea tener una efectividad del DRP de 0.96, la efectividad de la fase a mejorar debe alcanzar un valor $R_i = 0.83$, teniendo una mejora en la efectividad de esa fase de 57%.

En conclusión, el modelo de remoción de defectos se puede utilizar para calcular la efectividad de remoción actual, y la efectividad de remoción deseada, y nos permite decidir si se desea agregar una o más fases de remoción, o si se desea mejorar una o más fases de remoción existentes, calculando el efecto de cada caso en la efectividad del proceso de remoción de defectos general, o en las fases a mejorar.

3.4 Matriz de defectos.

Esta técnica utiliza una matriz para mostrar la distribución tanto de las inyecciones de defectos, como la distribución de detecciones y reparaciones de estos. Ejemplo:

		Etapa de inyección de defectos								
Etapa de remoción de defectos		REQ	HLD	DLD	CODE	UT	IT	ST	FIELD	Total
	REQ	0								0
	HLD	49	681							730
	DLD	6	42	681						729
	CODE	2	28	114	941					1095
	UT	21	43	43	223	2				332
	IT	0	41	61	261		4			387
	ST	6	8	24	72			1		111
	FIELD	8	16	16	40				1	81
	Total	122	859	939	1537	2	4	1	1	3465

Tabla 3.3 Matriz de defectos

3.4.1 Efectividad de remoción de defectos

La efectividad en la remoción de defectos por fase se define mediante lo siguiente:

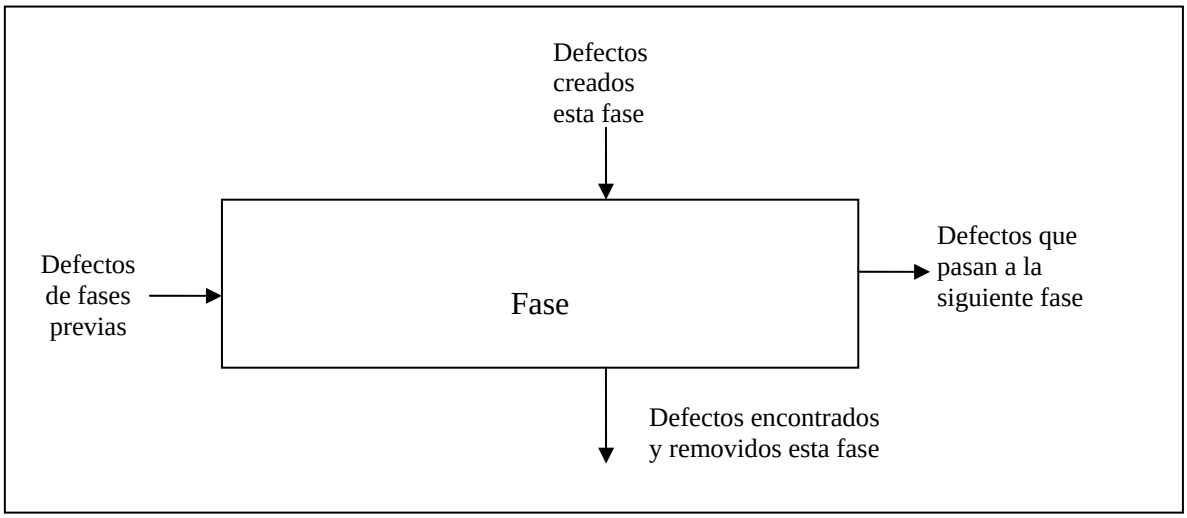

Figura 3.4 Defectos en una fase.

Defectos que pasan a la siguiente fase = (Previos + Creados) – Encontrados.

Efectividad E = 100 * Encontrados / (Previos + Creados).

Del ejemplo anterior:

Efectividad de remoción de defectos de la fase HLD

Defectos encontrados y removidos = 730
Defectos de la fase anterior (escapes) = 122
Defectos inyectados en HLD = 859
E(HLD) = 100 * (730) / (122 + 859) = 74%

Efectividad de remoción de defectos de la fase DLD

Defectos encontrados y removidos = 729
Defectos de la fase anterior (escapes) = 122+859-730 = 251
Defectos inyectados en DLD = 939
E(DLD) = 100 * (729) / (251 + 939) = 61%

Efectividad de remoción de defectos de la fase CODE

Defectos encontrados y removidos = 1095
Defectos de la fase anterior (escapes) = 122+859+939-730-729 = 461
Defectos inyectados en DLD = 1537
E(DLD) = 100 * (1095) / (461 + 1537) = 55%

De la misma manera se puede calcular la efectividad de las demás fases. Asimismo, es posible calcular la efectividad de las fases de diseño y codificación:

E = 100 * (730 + 729 +1095)/(122 + 859 + 939 + 1537) = 74%

Efectividad de todas las fases de pruebas:

E = 100 * (332+387+111)/(903 + 2 + 4 + 1) = 97.7%

4 Solución propuesta.

4.1 Procedimiento de clasificación y análisis

La solución propuesta consiste en lo siguiente:

a) Clasificar los defectos de acuerdo a los siguientes criterios:
- Por proyecto.
- Por tipo de componente.

b) Para cada uno de los conjuntos de defectos separados en las categorías antes descritas, se llevará a cabo lo siguiente:

- Se llenará una matriz de defectos como la establecida en la sección 3.4.

- Se calculará las métricas por cada fase del proceso de desarrollo:
  - Número de defectos inyectados (por fase y acumulativos).
  - Número de defectos detectados y removidos (por fase y acumulativos).
  - Número de defectos que escaparon a la detección (cantidad, y porcentaje de escape).
  - Efectividad de remoción de defectos (ERD).
  - Tiempo total de reparación de los defectos.
  - Porcentaje de reparación de los defectos por fase, respecto al tiempo total utilizado para reparar defectos en todo el proceso.
  - Porcentaje de reparación de los defectos inyectados en cada fase, respecto al tiempo total utilizado para reparar defectos en todo el proceso.

- En base a la matriz y a las métricas anteriores, se generarán gráficas:
  - Gráfica de número de defectos inyectados por fase.
  - Gráfica de número de defectos removidos por fase.
  - Para cada tipo de defecto con un número importante de ocurrencias, se generará una gráfica de número de inyecciones de ese tipo de defecto por fase, así como una gráfica de número de remociones de ese tipo de defecto por fase.

- Se observarán todos los datos de métricas y gráficas en conjunto, y junto con un análisis de las descripciones de los defectos, se identificará lo siguiente:
  - Fases del proceso con mayor inserción de defectos.
  - Fases dedicadas a inspecciones y pruebas, con posibles problemas de efectividad.
  - En base a las distribuciones de los distintos tipos de defectos a lo largo del proceso de desarrollo, se identificará las fases con más problemas de inyección de defectos.
  - Se propondrá mejoras a las fases con mayor inserción de defectos.

- Se propondrá mejoras para la efectividad de las fases dedicadas a la detección y remoción de defectos.

5 Análisis de datos

Se tomó una porción de los defectos, pertenecientes al proyecto 30, y se hizo el análisis propuesto. Se obtuvo lo siguiente:

5.1 Matriz de defectos.

Detecciones y remociones

Inserciones: Phase	HLD	DLD	DLDR	TD	DLDINS	CODE	CR	CODEINS	COMP	UT	IT	ST	Total found
REQ								1					1
HLD					3					1			4
ITP												1	1
DLD			150	1	144	26	19	20	19	61	6	17	463
DLDR													0
TD										7			7
DLDINS						2							2
CODE						1	97	52	47	50	2	4	253
CR													0
COMP										1			1
CODEINS													0
UT										3			3
IT													0
ST												1	1
													736

Tabla 5.1 Matriz de Defectos.

En base a las observaciones de la alta proporción de defectos en DLD y CODE se obtuvo las siguientes gráficas:

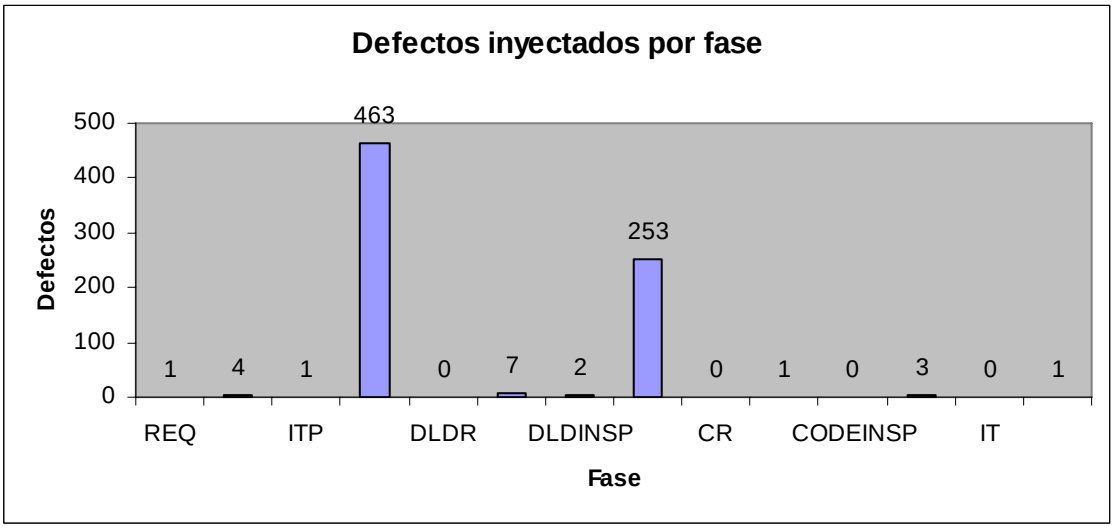

Fig. 5.2 Defectos inyectados por fase.

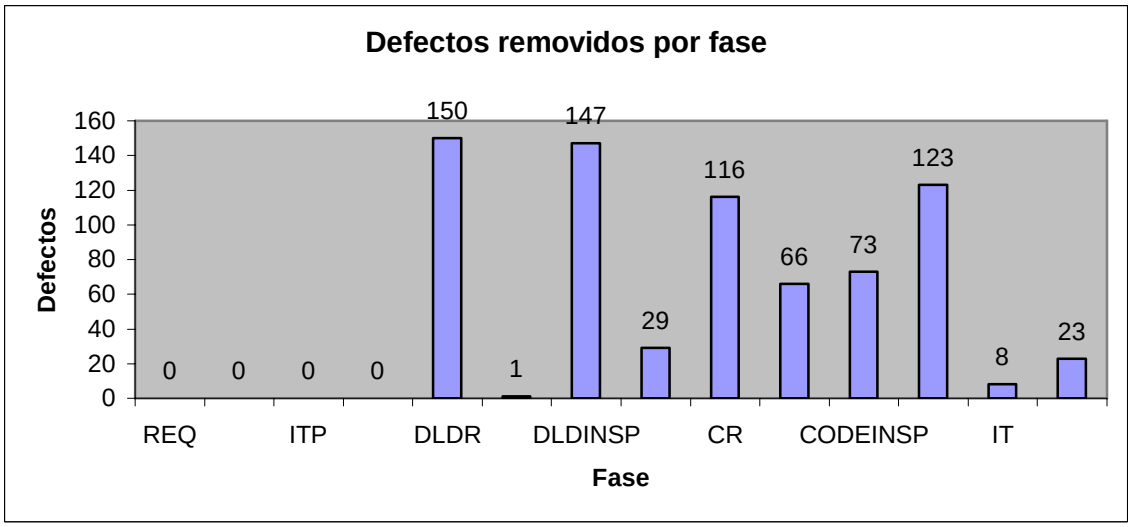

Fig. 5.3 Defectos removidos por fase.

De estas dos gráficas se empieza a observar problemas en el proceso de desarrollo, ya que si bien una buena parte de los errores insertados en DLD y CODE son encontrados en sus respectivas inspecciones y revisiones, también una cantidad considerable de ellos es encontrada en las etapas tardías del proceso de desarrollo (compilación, pruebas de unidad, pruebas de integración y pruebas de sistema).

Dado que los dos tipos principales de defectos son los insertados en las fases DLD y CODE, generamos las siguientes gráficas para saber dónde son encontrados y corregidos estos defectos:

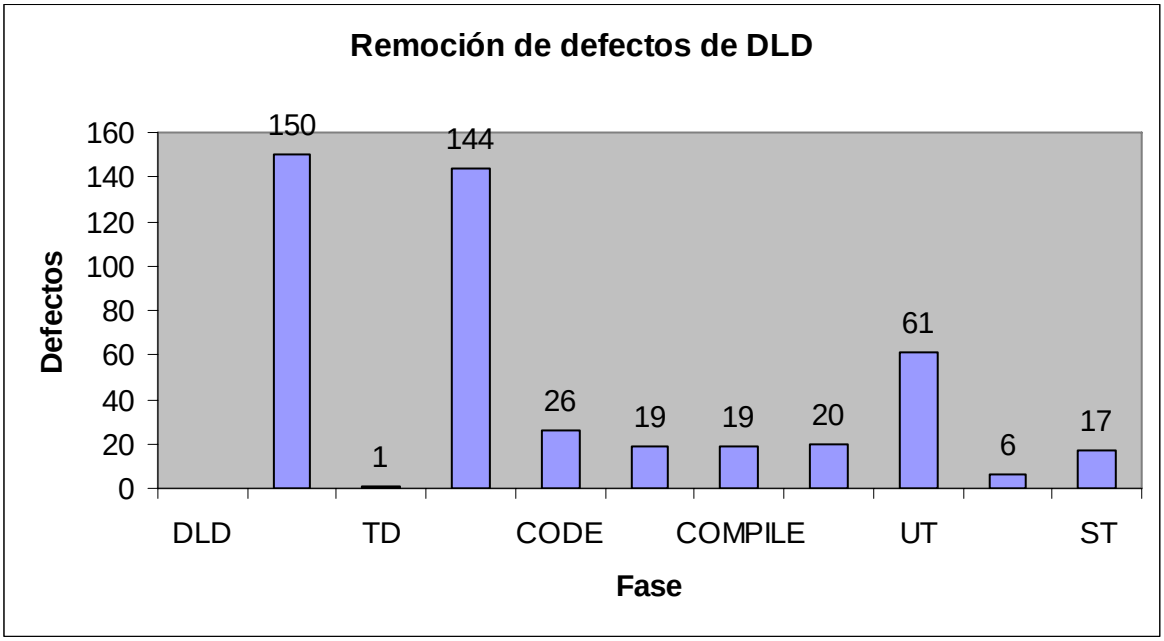

Fig. 5.4 Defectos de DLD removidos por fase.

De la gráfica anterior se observa que de los 463 defectos insertados en la fase DLD, 168 son encontrados después de la revisión e inspección de DLD. Esto significa que entre estas dos fases encuentran aproximadamente el 64% de los errores de este tipo, antes de pasar a fases que podríamos considerar tardías para encontrar estos defectos.

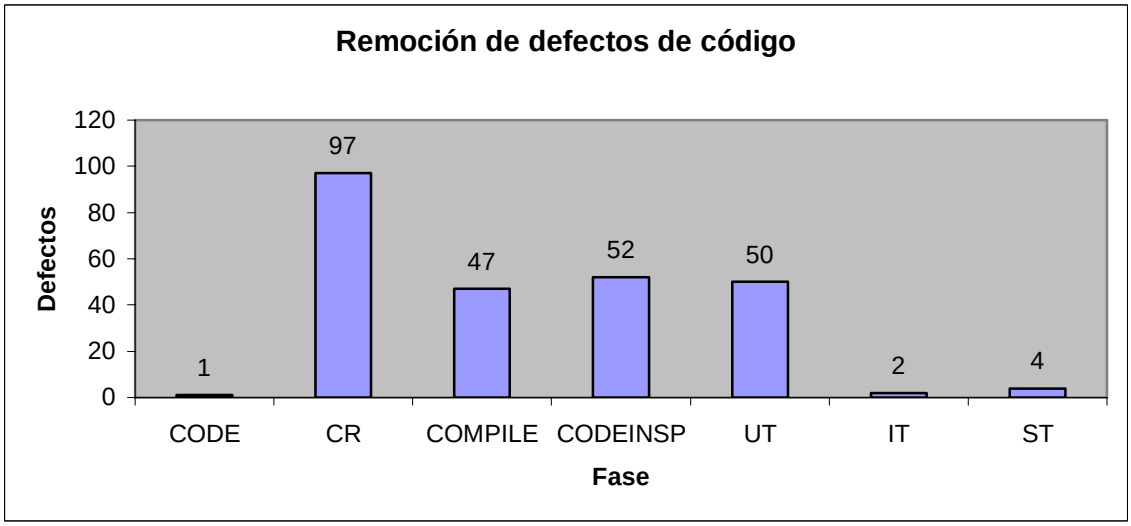

Fig. 5.5 Defectos de CODE removidos por fase.

De la gráfica anterior se observa que de los 253 defectos insertados en la fase DLD, 103 son encontrados después de la revisión e inspección de DLD. Esto significa que entre estas

dos fases encuentran aproximadamente el 60% de los errores de este tipo, antes de pasar a fases que podríamos considerar tardías para encontrar estos defectos.

A continuación se muestra la matriz de efectividad de detección y remoción de defectos resultante del proceso de desarrollo para este proceso:

Fase	inyectados	Detectados y borrados	Escapes	DRE	Tiempo total Reparación de defectos inyectados [min]	Porcentaje del tiempo total de reparación de los defectos inyectados	Tiempo total de reparación	Porcentaje de tiempo total de reparación
REQ	1	0	1	0	10.9	0.11009456		
HLD	4	0	5	0	64	0.64642677		
ITP	1	0	6	0	2	0.02020084		
DLD	463	0	469	0	6603.22	66.6952845		
DLDR	0	150	319	31.9829424			357.91	3.61504073
TD	7	1	325	0.30674847	62	0.62622594	22.1	0.22321925
DLDINS	2	147	180	44.9541284	3	0.03030126	720.52	7.27755344
CODE	253	29	404	6.69745958	3139.86	31.7138996	727.14	7.34441821
CR	0	116	288	28.7128713			514.42	5.19585721
COMP	1	66	223	22.8373702	5.33	0.05383523	884.67	8.93553711
CODEINS	0	73	150	32.735426			899.56	9.08593234
UT	3	123	30	80.3921569	5	0.05050209	4329.13	43.7260241
IT	0	8	22	26.6666667			320.1	3.23314392
ST	1	23	0	100	5.27	0.0532292	1125.03	11.3632737

Tabla. 5.2 Matriz de efectividad de detección y remoción de defectos.

Aquí se nota que las pruebas de unidad (UT) son muy efectivas, pero no lo son la revisión ni la inspección de diseño de bajo nivel (DLDR y DLDINS). Aquí también vemos que casi la mitad del tiempo total de reparación de defectos se realiza en la fase de pruebas de unidad, con 44%, seguida por el tiempo de reparaciones en pruebas de sistema, y compilación.

A continuación se muestra una tabla con los tipos de defecto encontrados, la cantidad de cada tipo de defecto, y los tiempos totales y promedio de reparación.

	Cantidad	Tiempo total	Tiempo promedio
Documentación	2	60	30
Estructura	13	17.5	1.34615385
Contenido Mayor	120	637.02	5.3085
Inconsistencia con requerimientos	3	18.8	6.26666667
Sintaxis	50	196.73	3.9346
Construcción	8	12.7	1.5875
Asignación	98	1362.36	13.9016327
Interfase	145	2464.28	16.9950345
Chequeo	43	486.57	11.3155814
Datos	8	26.7	3.3375
Función	238	4473.32	18.7954622
Ambiente	8	143.6	17.95

Tabla. 5.3 Matriz de tipos de defectos encontrados.

De la tabla anterior se observa que los tipos de defecto de función y de interfase son los tipos de defecto que más tiempo se necesitó para reparar.

A continuación se muestra la tabla de distribución de defectos de tipo Función a lo largo del proceso de desarrollo:

Errores de tipo Función	#	DLDR		DLDINS		CODE		CR		CODEINS		COMP		UT		ST		IT	
REQ	1									1	10.9								
HLD	4																		
ST	1			3	34									1	30	1	5.27		
DLD	127	26	133.7	30	183.93	9	66.1	12	232.54	12	247.21	3	95.92	26	1590	7	603.93	2	58.3
CODE	98							35	45.82	19	104	20	327	23	643.6	1	11.1		
TD	4													4	45				
UT	3													3					

Fig. 5.4 Tabla de distribución de defectos de tipo Función.

Y la tabla de distribución de defectos de tipo Interfase:

Errores de tipo Interfase		DLDR		DLDINS		CODE		CR		CODEINS		COMP		UT		ST		IT	
DLD	80	20	34.13	15	156	6	12	4	28.33	4	35	1	1	17	826	9	388	3	182
DLDINSP	2					2	3												
TD	1																		
CODE	62							23	62.14	15	154.5	5	14.1	15	446.7	3	97.12	1	1

Fig. 5.5 Tabla de distribución de defectos de tipo Interfase.

De esta tabla, junto con la información anterior se observa lo siguiente:

- Fases con más inserción de defectos: DLD y CODE.
- Tipos de defectos más comunes: Función, Interfase.
- Efectividad de detección muy alta de UT (80%)
- Efectividad de detección demasiado baja de DLDR (32%) y DLDINSP (45%)

- Los defectos en el diseño de tipo Función encontrados en etapas tardías (UT) tardan hasta 12 veces más en ser reparados que si se encuentran en etapas más inmediatas (DLDR, DLDINSP)
- Los defectos en el código de tipo Función encontrados en etapas tardías (UT) tardan hasta 22 veces más en ser reparados que si se encuentran en etapas más inmediatas (CR, CODEINSP)
- Los defectos en el diseño de tipo Interfase encontrados en etapas tardías (UT) tardan hasta 30 veces más en ser reparados que si se encuentran en etapas más inmediatas
- Los defectos en el código de tipo Interfase encontrados en etapas tardías (UT) tardan hasta 12 veces más en ser reparados que si se encuentran en etapas más inmediatas
- Se requiere aumentar la efectividad de las etapas de DLDR, DLDINSP, CR, y CODEINSP.

5.2 Recomendaciones específicas.

Para el caso de este proyecto, se generaron las siguiente recomendaciones específicas:

- Se observan muchos errores de codificación, tipo Función, en el los componentes Tapestry. Posiblemente mejorar la capacitación sobre Tapestry.
- Se observan muchos defectos de diseño debido a la utilización de funciones equivocadas, o falta de implementación de métodos. Se necesitan mejores técnicas de diseño. Agregar o mejorar en los checklists las revisiones de la correcta implementación y uso de los métodos.
- Se observan muchos errores de inicialización, asignación o comparación de valores. Hacer con mayor precaución el diseño detallado.
- Se observan muchos defectos de tipo Interfase, debidos a llamadas equivocadas a interfases, parámetros incorrectos, o llamadas a clases equivocadas. Hacer el diseño detallado con más cuidado. Posiblemente mayor capacitación sobre diseño.

6 Propuesta de cambios al proceso de desarrollo

Se propone incluir en el proceso de desarrollo actual lo siguiente:

- Revisar la clasificación actual de los defectos, y modificarla de manera que sea ortogonal.
- Realizar el procedimiento de análisis descrito en este trabajo, al final de cada etapa del proceso de desarrollo, con el fin de obtener conclusiones acerca de la efectividad de las tareas de aseguramiento de la calidad con que cuenta el proceso.
- Utilizando los datos obtenidos mediante el procedimiento anterior, llevar a cabo juntas de análisis causal después de cada fase en que se haya encontrado defectos en los distintos artefactos generados, con el fin de obtener lo más rápido posible datos acerca del desempeño del proceso de desarrollo, e implementar las acciones de prevención de defectos lo más temprano posible en el proceso, así como propagar estos cambios a otros proyectos de desarrollo lo más temprano posible.

7 Conclusiones

Durante la realización del presente trabajo, se llegó a una propuesta de procedimiento para analizar los datos registrados sobre los defectos inyectados y removidos durante un proceso de desarrollo.

7.1 Objetivos cumplidos.

El trabajo aquí presente cumplió con los siguientes objetivos:

- Proponer una forma de clasificación y análisis de datos de defectos.
- Generar métricas que describen el comportamiento de las actividades de aseguramiento de la calidad del proceso de desarrollo.
- En base a las métricas generadas, proponer cambios al proceso de desarrollo que permiten mejorar en gran medida la efectividad de las actividades de detección y remoción de los defectos, así como la prevención de los mismos durante las actividades de desarrollo.
- Proponer la inserción en el método de desarrollo actual, de actividades altamente efectivas de aseguramiento de la calidad, como son el proceso de prevención de defectos, análisis causal de defectos, y la clasificación ortogonal de defectos.

7.2 Contribuciones de este trabajo.

- Se justificó el método de clasificación y análisis utilizado en este documento.
- Se propuso la utilización de ciertas métricas para conocer el grado de aseguramiento de la calidad del proceso de desarrollo.
- Se explicó el método propuesto con un ejemplo.
- Se generaron recomendaciones específicas para cada proyecto, y recomendaciones generales para el proceso general de desarrollo actual.

8 Referencias

Humphrey, W. "*A Personal Commitment to Software Quality*" The Software Engineering Institute. Carnegie Mellon University. Pittsburgh, PA.

Mandeville, W. "*Software Costs of Quality*" IEEE Journal on Selected Areas in Communications, Vol 8. No. 2. February 1990. 0733-8716/90/0200-0315

Fredericks, M. Basili, V. "*Using Defect Tracking and Analysis to Improve Software Quality*" DoD Data & Analysis Center for Software (DACS). ITT Industries – Systems Division. Rome, NY. Prepared for: Air Force Research Laboratory – Information Directorate (AFRL/IF). Contract Number: SP0700-98-D-4000.

Chillarege, R. Bhandari, I. Chaar, K, Halliday, M. Moebus, D. Ray, B. Wong, M. "*Orthogonal Defect Classification – A Concept for In-Process Measurements*". IEEE Transactions on Software Engineering, Vol 18, No. 11. November 1992. 0162-8828/92. IEEE

Chillarege, R. Kao, W. Condit, R. "Defect *Type and its Impact on the Growth Curve*". IBM Thomas J. Watson Research Center. Yorktown Heights, NY. CH2892-7/91/0000/0246 IEEE.

Bhandari, I. Halliday, M.J. Chaar, J. Chillarege, R. Jones, J. Atkinson, J.S. Lepori-Costello, C. Jasper, P.Y. Tarver, E.D. Lewis, C.C, Yonezawa, M. "*In-process improvement through defect data interpretation*" IBM Systems Journal, Vol 33. No1. 1994

Leung, H. "*Improving Defect Removal Effectiveness for Software Development*". Department of Computing. The Hong Kong Polytechnic University.

Fagan, M. "*Advances in Software Inspections*". IEEE Transactions in Software Engineering. Vol. 12, No. 7, July 1986.

Jose, A. Pillai, S. Anju, N. "Closed *loop defect removal model using statistical process control*". Quality Engineering Group, NeST, Trivandrum.

Booker, G. "Evaluation *of Information Systems – Defect Analysis and Removal*".

Mays. R. "*Applications of Defect Prevention in Software Development*". IEEE Journal on Selected Areas in Communications. Vol 8, No 2. February 1990. 0733-8716/90/0200-0164.

Gilb. T. Graham, D. "*Software Inspection*". Chapter 17: "*Practical Aspects of the Defect Prevention Process*". Addison-Wesley, 1993.