REPORTE DE ANILISIS DE DEFECTOS “R3”

PROCESO PERSONAL DE SOFTWARE

JUAN ALBERTO GUTIERREZ CANTO MATRICULA:2440063

2016

Tabla de contenido

[OBJETIVOS 2](#_Toc444621200)

[ANALISIS DE DEFECTOS 2](#_Toc444621201)

[TOTAL, DE DEFECTOS POR PROGRAMA 2](#_Toc444621202)

[DEFECTOS INYECTADOS EN DISEÑO POR PROGRAMA 3](#_Toc444621203)

[DEFECTOS INYECTADOS EN CODIFICACION POR PROGRAMA 5](#_Toc444621204)

[DEFECTOS REMOVIDOS EN DISEÑO POR PROGRAMA 6](#_Toc444621205)

[DEFECTOS REMOVIDOS EN CODIFICACION POR PROGRAMA 7](#_Toc444621206)

[DEFECTOS REMOVIDOS EN COPILACION POR PROGRAMA 8](#_Toc444621207)

[DEFECTOS REMOVIDOS EN PRUEBAS POR PROGRAMA 9](#_Toc444621208)

[DEFECTOS INYECTADOS A LA FECHA POR FASE 11](#_Toc444621209)

[DEFECTOS REMOVIDOS A LA FECHA POR FASE 12](#_Toc444621210)

[ANALISIS DE DEFECTOS MEDIANTE PARETO 13](#_Toc444621211)

[DEFECTOS REMOVIDOS POR TIPO 13](#_Toc444621212)

[TIEMPO DE CORRECCIONES DE DEFECTOS POR TIPO 14](#_Toc444621213)

[TABLAS DE ANALISIS DE DATOS 15](#_Toc444621214)

[DENSIDAD DE LOS DEFECTOS 15](#_Toc444621215)

[TIEMPO DE CORRECCIONES DE DEFECTOS 16](#_Toc444621216)

[CONCLUCION 17](#_Toc444621217)

# OBJETIVOS

El reporte R3 es generado para entender la densidad y los tipos de defectos que son introducidos dentro del desarrollo de los programas 1A, 2A y 3A.

También muestra la importancia de tener almacenados en una base de datos, todos los datos de los defectos introducidos.

# ANALISIS DE DEFECTOS

Dentro del desarrollo de los programas hasta el momento, se ha generado una cantidad constantes de errores.

Estos pueden ser por diferentes cuestiones y, o formas de hacer el proceso de desarrollo, entraremos más a detalle acerca de que son los errores, cuales son los errores y porque se cometen estos errores durante el proceso de creación de software.

## TOTAL, DE DEFECTOS POR PROGRAMA

El total de defectos por programa nos denota una interacción en cuanto a Defects/KLOC y el número del programa realizado.

Lo que podemos notar de la gráfica es lo siguiente:

* Existe una gran cantidad de defectos que se inyectan en la fase de creación del software
* Entre mayor es el código existen más defectos que se inyectan
* El tiempo de corrección de errores es muy alto
* Se destina un 30% del proyecto a la corrección de errores
* La cantidad de errores inyectados después del proyecto aún es grande para el proyecto.

## DEFECTOS INYECTADOS EN DISEÑO POR PROGRAMA

Defectos inyectados en diseño por programa, como podemos ver es una relación en cuanto a defects/KLOC contra el numero del programa.

Lo que podemos ver en la gráfica es lo siguiente:

* El programa 1A no muestra una tasa de errores en código, esto se dio por que en el programa 1ª no se tuvo una fase concreta de diseño, esto quiere decir que el diseño en general del programa se realizó en el código.
* En el programa que más defectos se han introducido en la parte de diseño es en el 2A.
* Parte de que los defectos se encontraron más en el programa 2A, fue debido a que el programa tuvo una falla en su estructura funcional principal, la cual no se tomó en cuenta en la parte de diseño, simplemente se pasó por alto, esto provoco que se inyectaran más defectos al intentar corregirla y cambiar varas funciones del programa mismo.
* En el programa 3 A no se generó muchos errores dentro de la fase del diseño, esto fue debido a que el programa 3 A fue una ampliación del programa 2 A, esto facilito el diseño y solamente se le agregaron funcionalidades nuevas.

## DEFECTOS INYECTADOS EN CODIFICACION POR PROGRAMA

Defectos inyectados en código por programa, la gráfica anterior nos representa una relación entre defects/KLOC y numero del programa.

Lo que podemos ver en ella es:

* En la gráfica observamos que no tenemos errores insertados en el programa 1 A.
* Del programa en los que tenemos más errores insertados en el programa 2 A, debido a que representa uno de los códigos que más líneas funcionales contiene y por lo tanto genera una mayor tasa de errores, además de que varios de estos errores se causaron por la utilización de funciones nuevas, lo que también provoco que los elementos enviados a estas fallaran. Se podría definir que la utilización de funciones no conocidas genera una mayor oportunidad a cometer errores, por valores mal referenciados o por una mala utilización.
* En cuanto a los erres que se inyectan más en código, tenemos los de asignación, esto es debido a que no se piensa en la utilización de variables como contadores o valores que tienen que estar definidos en cierto momento de la aplicación.
* Otros errores que se insertan mucho en codificación son los de funcionalidad, ya que no se está pensando en el diseño y en dos de los tres programas hechos al momento se está diseñando aun en codificación esto genera muchos errores en esta fase.

## DEFECTOS REMOVIDOS EN DISEÑO POR PROGRAMA

En la gráfica defectos removidos en diseño que se muestra en la parte de arriba, existe una relación entre defects/KLOC y numero de programa.

En ella podemos ver:

* El número de líneas de código que se han removido en la parte de diseño
* Debido a que no se ha realizado una buena fase de diseño, no se remueve ningún defecto en esta fase.
* La falta de diseño provoca una gran cantidad de defectos en el código.
* Si se empezara a diseñar más en esta fase y no en codificación, se generaría un producto con una mejor calidad y se aprovecharía al máximo las fases de diseño y codificación.

## DEFECTOS REMOVIDOS EN CODIFICACION POR PROGRAMA

En la gráfica anterior podemos ver la relación entre defects/loc y el número de programa acerca del número de defectos removidos en fase de codificación.

El grafica podemos observar que:

* Existe una nula cantidad de defectos que se remueven en la fase de codificación.
* El error acerca de que no existan defectos en esta parte del código puede sugerir dos cosas:
  + La fase de codificación se está tomando demasiado tiempo en diseñar en código la funcionalidad del programa, esto causa que solamente se plante el diseño en código y se ejecute y no se utilice una buena práctica de codificación.
  + No existe una buena práctica de codificación, esto lo referimos más a no solamente codificar, si no imprimirle un poco más de esfuerzo dentro de la fase. Se puede no solamente codificar y pasar a la siguiente fase, sería bueno revisar el código una vez terminado, para eliminar gran parte de los errores que aparecen en compilación, que son los errores de sintaxis. O eliminar errores que se tiene en diseño, o eliminar todas esas dudas acerca de que va a estar mal.

## DEFECTOS REMOVIDOS EN COPILACION POR PROGRAMA

En la gráfica de defectos removidos en compilación podemos observar la relacionante defect/KLOC y numero de programas.

De la gráfica también podemos observar que:

* Una vez más tendemos a que el programa 2 A tiene más errores en la fase.
* De las grandes cosas que podemos observar y darnos cuenta esta el hecho de que la mayoría de los errores que se encuentran en esta fase son de sintaxis. Esto es debido a una falta de buena práctica en la parte de codificación.
* A pesar de que no se tarda mucho tiempo corrigiendo errores en esta fase, son muchos los errores pequeños que se encuentran aquí.

## DEFECTOS REMOVIDOS EN PRUEBAS POR PROGRAMA

En la gráfica podemos observar los defectos corregidos en la fase de pruebas, está dada por la relación defects/KLOC y el número del programa.

También podemos observar que:

* Las pruebas están dando más errores que otras fases
* En esta fase es donde más tiempo tardamos corrigiendo errores
* La mayoría de los errores encontrados en esta fase se cargan a el código y al diseño
* Una de las formas como podríamos eliminar gran parte de los defectos que aparecen en esta parte, puede ser la planeación y diseño de las pruebas en la parte de diseño. Ya que así se puede pensar concretamente en todo el diseño del programa y hacer una mejor validación para el código.
* El código presenta muchos errores de datos, se asignan muchos valores erróneos a las funciones, formulas y procedimientos. Esto se puede evitar diseñando y revisando desde antes la documentación del lenguaje, para así obtener lo mejor de esta parte.

## DEFECTOS INYECTADOS A LA FECHA POR FASE

La grafica siguiente nos presenta los defectos introducidos por fases, en la grafica se encuantra una relacion %defects con el numero de programa.

Tambien podemos observar lo siguiente:

* Las faces en donde se inyectan la mayor cantidad de errores están el código y el diseño
* A pesar de que no se tiene un diseño bien hecho, el poco diseño está causando una tasa de erros alta
* La parte de codificación tiene la tasa más elevada de inyección de errores, más del 50 % de los errores se inyectan en código.
* El programa 2 A tiene la mayor cantidad de errores inyectados e las dos fases lo que nos indica que aunque el código tenga más líneas, la tasa de errores y el crecimiento de estos se mantiene constante, aun se inyecta la misma proporción de errores sea cual sea su tamaño.

## DEFECTOS REMOVIDOS A LA FECHA POR FASE

La grafica nos representa los defectos removidos por fase, esta se encuentra en una relación de % defects y numero de programa.

También podemos observar que:

* La mayor cantidad de defectos son removidos en la parte de test, ya que el código está bien escrito la mayoría de veces, pero los problemas de funcionalidad son brutales. Se está fallando muchísimo el programa en dar las pruebas correctas, esto es parte de diseño, no se puede tener buenos resultados si no se diseña correctamente, si no se tiene un buen diseño siempre se va a fallar.
* De los problemas que también observamos en mayor cantidad, son problemas de documentación, pequeños problemas que se corrigen rápido pero que son vitales para tener un buen proceso de PSP.
* Al analizar los problemas que encontramos con mayor frecuencia nos podemos dar cuanta que son problemas que se tiene por no dar una segunda revisión a los productos entregados.

# ANALISIS DE DEFECTOS MEDIANTE PARETO

## DEFECTOS REMOVIDOS POR TIPO

En la tabla podemos observar los errores removidos pro tipo, están en una relación cantidad de errores con tipo de errores.

También podemos observar que:

* La mayor cantidad de errores que se tiene son en cuanto a problemas de documentación o archivos que no tiene funcionalidad del proyecto, pero si tiene un gran peso a la administración y entendimiento del proceso.
* La segunda cantidad de errores introducidos son de funcionalidad, esto nos lleva a querer mejorar nuestro diseño y pensar mejor que es lo que se tiene que hacer y cuando se tiene que hacer, asignar una mejor descripción y formatos a llenar de las fases del desarrollo, se está codificando mejor que diseñando y no hay una buena función sin un buen diseño.
* Las asignaciones dentro del código presentan una gran cantidad también de problemas encontrados, falta poner mayor atención a las revisiones.
* Por el momento no se tiene muchos problemas en cuanto a la funcionalidad del ambiente en cuanto se está trabajando o cuanto a las aplicaciones funcionales para hacer el proyecto.

## TIEMPO DE CORRECCIONES DE DEFECTOS POR TIPO

En esta grafica podemos observar el tiempo tardado en corregir los errores por tipo, podemos ver la relación tiempo en minutos con el tipo de errores.

También podemos observar que:

* La mayor cantidad de errores está en documentación y nos causa mucho tiempo en corrección de estos
* El error de funcionalidad nos está llevando mucho tiempo corregirlos y al momento están inyectando nuevos errores y generan más tiempo.
* Los errores de codificación en cuanto a sintaxis, a pesar de que son más se corrigen en muy poco tiempo
* Se está tomando aproximadamente una hora más en el proyecto por corregir errores.

# TABLAS DE ANALISIS DE DATOS

## DENSIDAD DE LOS DEFECTOS

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Defect Densities** | | | | **Compile and Test Defects** | | | |
| Program Number | New and Changed LOC | Total Defects | Defects per KLOC | Defects found in compile | Compile defects per KLOC | Defects found in test | Test defects per KLOC |
| 1 | 0 | 6 | 0 | 1 | 0 | 2 | 0 |
| 2 | 26 | 5 | 192 | 1 | 38 | 4 | 154 |
| 3 | 133 | 5 | 38 | 3 | 23 | 2 | 15 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Totals | 159 | 16 | 101 | 5 | 31 | 8 | 50 |

En la tabla podemos observar la densidad de los defectos que se han inyectado, así como las faces en donde se están inyectando.

Dos de las fases que se toman en cuenta son diseño, compilación y pruebas.

Podemos observar que la mayoría de los problemas se encuentran en estas fases y se remueven en pruebas.

Se puede observar que una gran cantidad de KLOCs se están modificando por la cantidad de errores introducidos.

La planeación de los proyectos se está cargando más en tiempos de pruebas y compilación, debido a que existe muchos problemas que se corrigen en estas fases y provocan excedentes en el tiempo del proyecto.

De las fases donde más se introducen códigos y no se tiene mucho tiempo empleado, es en la parte de diseño, no se diseña correctamente. Está faltando mucho en el diseño, así como en el diseño de pruebas.

En cuanto a el proceso de PSP la mayor cantidad de problemas de administración se encuentran en la falta de revisión de los productos, se están entregando con muy poco tiempo y sin calidad.

## TIEMPO DE CORRECCIONES DE DEFECTOS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Defect Fix Times** | | | | |
|  |  | Defects found in compiling | Defects found in testing | Total defects found |
| Defects | Tot. fix time | 0.00 | 8.00 | 11.00 |
| injected in | Tot. defects | 0.00 | 3.00 | 4.00 |
| designing | Avg. fix time | 0.00 | 2.67 | 2.75 |
| Defects | Tot. fix time | 5.00 | 14.00 | 20.00 |
| injected in | Tot. defects | 5.00 | 4.00 | 10.00 |
| coding | Avg. fix time | 1.00 | 3.50 | 2.00 |
| Total | Tot. fix time | 5.00 | 24.00 | 51.00 |
| defects | Tot. defects | 5.00 | 8.00 | 21.00 |
| injected | Avg. fix time | 1.00 | 3.00 | 2.43 |

El reporte de tiempo de corrección de defectos lo podemos definir como que:

Podemos ver claramente que la mayor cantidad de defectos es inyectada en fases de diseño y codificación, la mayoría son removidos en pruebas. Pero la otra mitad de errores son de post mortem, con el proyecto finalizado, se están teniendo problemas de corrección de documentación muy recurrentemente.

Existe un tiempo muy grande en el cual se corrigen errores y esto no está del todo correcto, se tiene que estimar una gran cantidad de tiempo para la corrección de errores dentro de los proyectos.

# CONCLUCION

Podemos decir al final del reporte R3 que una gran cantidad de errores se están inyectando en fases de desarrollo del proyecto, esto quiere decir que no se está haciendo bien una fase del proyecto que nos está llevando a introducir muchos errores en fases de creación.

Eh notado que muchos de los errores se deben a la falta de diseño, el diseño que se está haciendo es algo muy básico y sin forma, no se puede entender todas las funcionalidades del proyecto. Si esta parte falla, genera una impresionante cantidad de errores en las siguientes fases.

También eh notado que hace falta imprimir un poco de calidad a los proyectos, esto quiere decir, que se deben de checar una segunda vez antes de dar por terminado el proyecto.