INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES CAMPUS MONTERREY

CALIDAD Y PRUEBAS DE SOFTWARE

TC3045

ING. RAFAEL SALAZAR CHÁVEZ

HERNÁN IRUEGAS VILLARREAL

A00817021

GRUPO: 1

05/05/2018

Índice

Introducción 4

Análisis de la Estimación de Tamaño 4

1. ¿Cuál fue el tamaño (LDC AyM) de cada uno de tus programas? ¿Cuál fue tu promedio de LDC AyM desarrolladas por programa? 4

2. ¿Cuál fue el programa con más LDC AyM y cuál el que tuvo menos? ¿Por qué? 5

3. ¿Cuál fue el % de error en la estimación de tamaño (real/plan-1) en tus programas 2 al 7? En promedio, ¿Cuál fue el % de error en la estimación de tamaño de estos 6 programas juntos? 5

4. ¿Cuál fue el programa con mayor error de estimación de tamaño y cuál el menor? ¿Por qué? 6

5. En aquellos programas donde pudiste utilizar el método PROBE “A” o el “B”, ¿el tamaño real de LDC AyM estuvo dentro del intervalo de predicción estadístico del 70%? Si no estuvo, ¿Qué pasó? 6

6. Basado en los datos históricos que has analizado sobre el error de estimación del tamaño, ¿Qué meta realista (alcanzable) te puedes poner para mejorar? 6

7. ¿Qué cambios vas a hacerle a tu proceso personal para alcanzar esta meta? 7

Análisis de la Estimación de Tiempo 7

1. ¿Cuál fue el tiempo en horas dedicado a cada uno de tus programas? ¿Cuál fue tu promedio de horas dedicadas por programa? 7

2. ¿Cuál fue el programa al que le dedicaste más tiempo y a cuál menos? ¿Por qué? 7

3. ¿Cuál fue el % de error en la estimación de tiempo (real/plan-1) en tus programas 1 al 7? En promedio, ¿Cuál fue el % de error en la estimación de tiempo de estos 7 programas juntos? 8

4. ¿Cuál fue el programa con mayor error de estimación de tiempo y cuál el menor? ¿Por qué? 9

5. En aquellos programas donde pudiste utilizar el método PROBE “A” o el “B”, ¿El tiempo real estuvo dentro del intervalo de predicción estadístico del 70%? Si no estuvo, ¿Qué pasó? 10

6. ¿Cuál fue la productividad de cada uno de tus programas? ¿Cuál fue tu productividad promedio? 10

7. ¿Cuál es la tendencia de tu productividad? ¿Va a la baja, o al alta? ¿Es inestable? ¿Qué puedes hacer para mejorarla? 11

8. ¿Qué tanto fueron afectados los errores de estimación de tiempo por tus errores de estimación del tamaño? 12

9. Basado en los datos históricos que has analizado sobre el error de estimación del tiempo, ¿Qué meta realista (alcanzable) te puedes poner para mejorar? 14

10. ¿Qué cambios vas a hacerle a tu proceso personal para alcanzar esta meta? 14

Análisis de la Calidad (Defectos) 14

1. ¿Qué tipo de defecto es al que más tiempo le dedicaste en la fase de compilación? ¿Y en la fase de pruebas? 14

2. ¿Qué tipo de defecto es el que más inyectaste en la fase de Diseño? ¿Y en la fase de Codificación? 15

3. ¿Cuál es el promedio de horas por defecto dedicadas al remover defectos en las fases de Revisión del Diseño, Revisión del Código, Compilación y Pruebas? 15

4. ¿Cuál es la tendencia en los 7 programas del total de defectos inyectados por KLDC? 16

5. ¿Cuál es la tendencia en los 7 programas de los defectos removidos por KLDC en las fases de Revisión del Diseño, Revisión del Código, Compilación y Pruebas? 16

6. ¿Cómo se comparan las tasas de remoción de defectos (defectos removidos por hora) en las fases de Revisión del Diseño, Revisión del Código, Compilación y Pruebas en los 7 programas? 18

7. ¿Cuál fue tu velocidad (LDC por hora) a la que revisaste el código de cada programa? 18

8. ¿Cuál fue la palanca de remoción de defectos (DRL) de cada programa para las fases de Revisión del Diseño y Revisión del Código? ¿Es bueno o malo? 19

9. ¿Existe alguna relación entre el Yield de cada programa y la velocidad (LDC por hora) a la que revisaste el código? 20

10. ¿Existe alguna relación entre el Yield de cada programa y su A/FR? 20

11. ¿Estás encontrando los defectos en las fases de Revisión del Diseño y del Código, o se te están escapando a las fases de Compilación y Pruebas? ¿Por qué? 21

12. Basado en los datos históricos que has analizado sobre la calidad de tus programas, ¿Qué meta realista (alcanzable) te puedes poner para mejorar? 23

13. ¿Qué cambios vas a hacerle a tu proceso personal para alcanzar esta meta? 23

Conclusión: 23

Párrafo de honestidad: 24

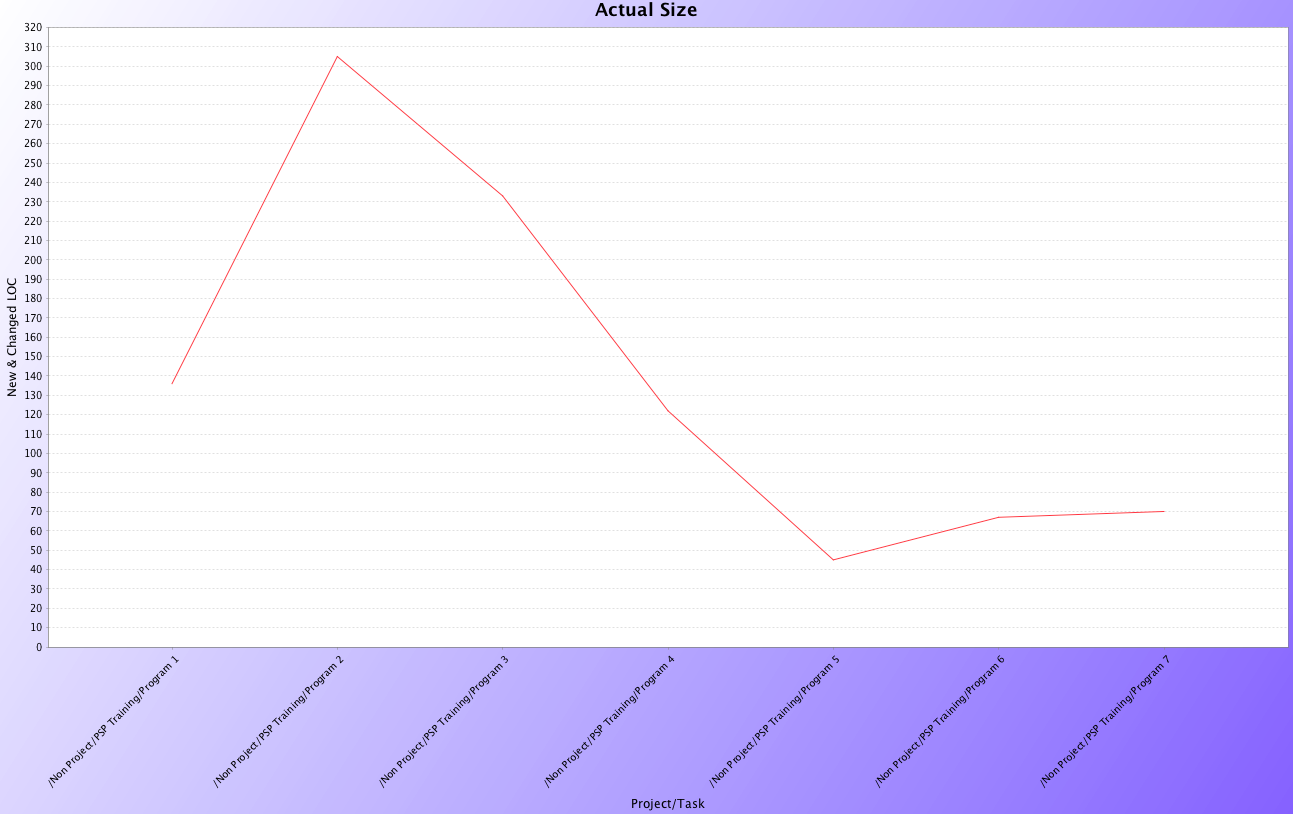
# Introducción

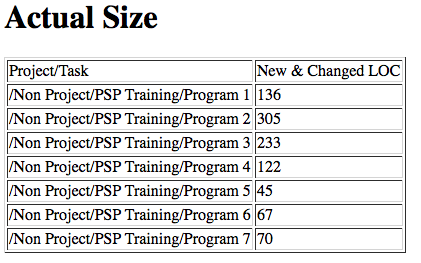
Este reporte tiene el propósito de mostrar el rendimiento de un programador a lo largo del desarrollo de 7 programas, diseñados para obtener metricas comparables que el desarrollador pueda utilizar para mejorar su proceso personal y así conocer sus fuerzas y debilidades. Se proveeran grafícos para ilustrar las estadísticas que se quieran demostrar, habra un pequeño análisis general que justifique y explique el significado de dichos gráficos, tablas o estadísticas. Todas las metricas obtenidas en este reporte se generaron utilizando un proceso incremental del PSP, por lo que son comparables y relevantes.

# Análisis de la Estimación de Tamaño

## ¿Cuál fue el tamaño (LDC AyM) de cada uno de tus programas? ¿Cuál fue tu promedio de LDC AyM desarrolladas por programa?

Tamaño (LDC AyM) de cada programa:





La razón por la que la gráfica tiene una tendencia a disminuir en cuanto a LDC AyM se debe a la reutilización de código, al decremento en dificultad de los programas a partir del programa 2, y a la similitud de los requerimientos a partir del programa 4.

Promedio (LDC AyM):

* 139.71428571

## ¿Cuál fue el programa con más LDC AyM y cuál el que tuvo menos? ¿Por qué?

Programa con más (LDC AyM):

* Programa 2 con 305

Programa con menos (LDC AyM):

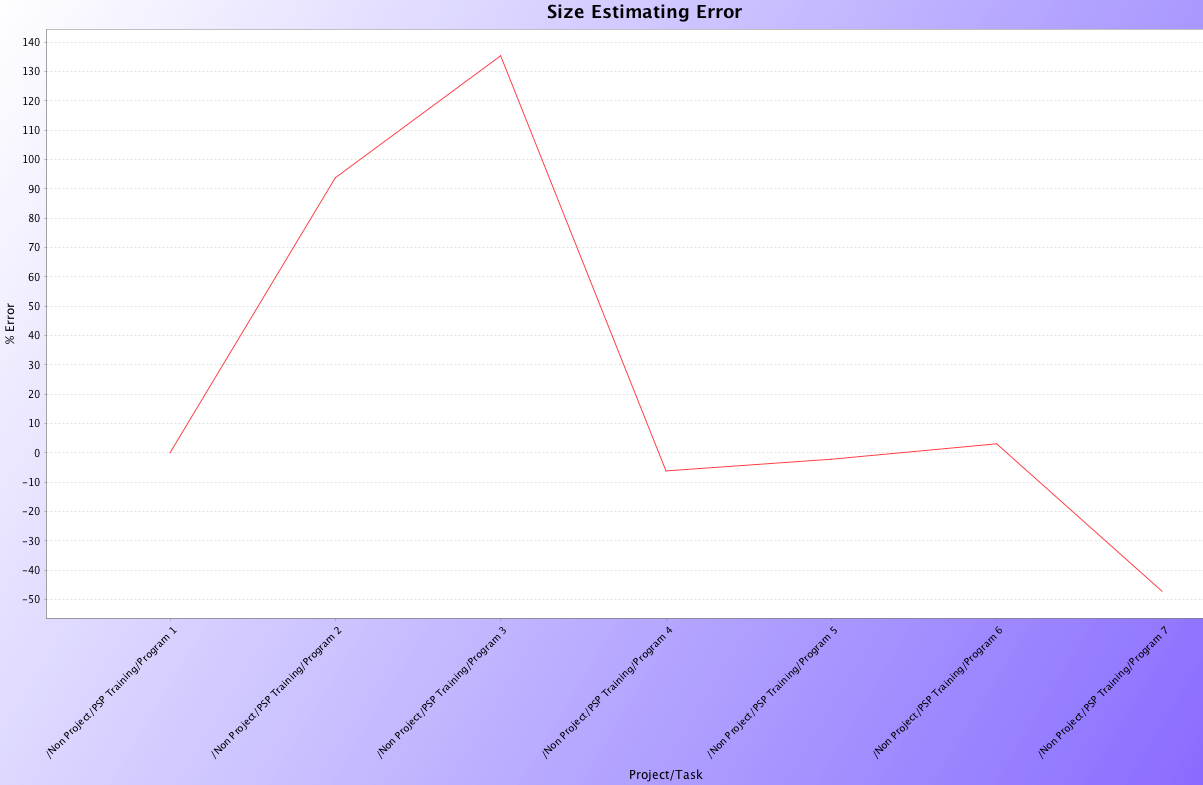
* Programa 5 con 45

La razón por la que el programa 2 tuvo más (LDC AyM) fue debido a la dificultad (programa con mayor dificultad de todos), poca reutilización de código debido a que fue el programa con la menor similitud de requerimientos con los demás. En otras palabrás, el programa 2 cumplía una función diferente y no está diseñado para añadir funcionalidad posteriormente.

El programa 5 fue el de menor (LDC AyM) debido a la reutilización de código proveniente de los programas anteriores, a la similitud en requerimientos y funcionalidades con los programas anteriores (excepto el 2) y al nivel de dificultad (uno de los programas más sencillos).

## ¿Cuál fue el % de error en la estimación de tamaño (real/plan-1) en tus programas 2 al 7? En promedio, ¿Cuál fue el % de error en la estimación de tamaño de estos 6 programas juntos?

Porcentaje de error en la estimación de tamaño (real/plan) en programas 2 al 7:

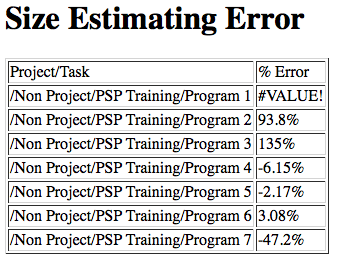


Promedio % de error en la estimación de tamaño de estos 6 programas juntos:

* 47.9%

La razón por la que tiende a disminuir el % de error en los programas es debido a el historial de datos en programas anteriores y la mejora del PSP personal en cuanto a revisiones y estimaciones.

## ¿Cuál fue el programa con mayor error de estimación de tamaño y cuál el menor? ¿Por qué?



Programa con mayor error de estimación de tamaño:

* Programa 3 con 135%

Programa con menor error de estimación de tamaño:

* Programa 5 con 2.17%

La razón por la que el programa 2 tiene el mayor error de estimación de tamaño se debe a que no se tenían suficientes datos históricos para poder realizar estimaciones, por lo que la estimación se genero con un “juicio personal” sin experiencia.

La razón por la que el programa 5 tiene el menor error de estimación de tamaño se debe a que se contaban con datos históricos y experiencia suficiente para realizar estimaciones adecuadas. Además, este programa resulto ser uno de los más sencillos (de hecho fue el de menor LDC AyM) por lo que los requerimientos se entendieron de mejor manera y por ello se pudo estimar lo que se necesitaba de manera más eficaz.

## En aquellos programas donde pudiste utilizar el método PROBE “A” o el “B”, ¿el tamaño real de LDC AyM estuvo dentro del intervalo de predicción estadístico del 70%? Si no estuvo, ¿Qué pasó?

No se utilizaron los métodos A o B puesto que nunca fueron los más recomendados por Process Dashboard.

## Basado en los datos históricos que has analizado sobre el error de estimación del tamaño, ¿Qué meta realista (alcanzable) te puedes poner para mejorar?

Tener un seguimiento de los porcentajes de error en los programas de mi vida profesional para calcular un porcentaje de error que sirva como “colchon”, el cual sera modificable de acuerdo a programas realizados recientemente y experiencia ganada en la tarea de estimar tamaños. Esto servirá para disminuir el margen de error promedio en los programas futuros para llegar a un margen de error de 20%.

## ¿Qué cambios vas a hacerle a tu proceso personal para alcanzar esta meta?

Calcular de acuerdo a peores escenarios posibles, con el fin de tener un “colchon” y no subestimar al tamaño final del programa considerablemente.

Se debe considerar agregar al tamaño del diseño, un porcentaje de margen de error. Este porcentaje debe ser algo apropiado teniendo en cuenta los datos históricos y experiencia personal. Es importante considerar que este margen de error debe ser modificable de acuerdo con la experiencia y mejores predicciones en la estimación de tamaño. Por ejemplo, sería prudente comenzar con un “colchon” de 47.9% para la estimación de tamaño puesto que este fue el promedio obtenido de los programas 2 al 7. No obstante, conforme las predicciones mejoran y el porcentaje de error disminuya, lo mejor debe ser disminuir el “colchon” para que refleje la mejora en la predicción del tamaño de acuerdo a la experiencia.

# Análisis de la Estimación de Tiempo

## ¿Cuál fue el tiempo en horas dedicado a cada uno de tus programas? ¿Cuál fue tu promedio de horas dedicadas por programa?

Tiempo de cada programa:

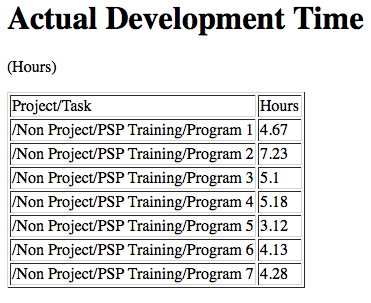


Promedio de horas:

* 4.8157142857

La razón por la que la gráfica tiene una tendencia a disminuir en cuanto a tiempo dedidcado por programa se debe a la reutilización de código, al decremento en dificultad de los programas a partir del programa 2, y a la similitud de los requerimientos a partir del programa 4.

## ¿Cuál fue el programa al que le dedicaste más tiempo y a cuál menos? ¿Por qué?



Programa con el mayor numero de horas:

* Programa 2 con 7.23

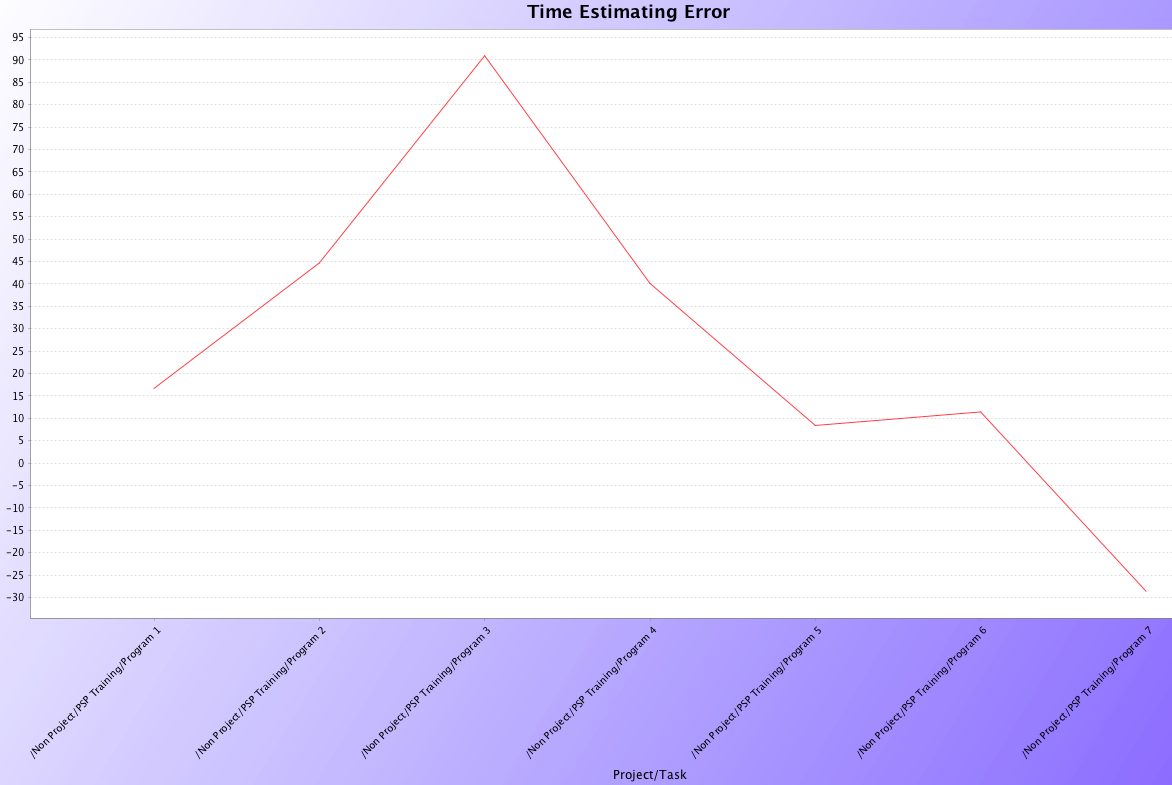
Programa con el menor numero de horas:

* Programa 5 con 3.12

La razón por la que el programa 2 tuvo más horas fue debido a la dificultad (programa con mayor dificultad de todos), poca reutilización de código debido a que fue el programa con la menor similitud de requerimientos con los demás. En otras palabrás, el programa 2 cumplía una función diferente y no está diseñado para añadir funcionalidad posteriormente.

El programa 5 fue el de menor horas debido a la reutilización de código proveniente de los programas anteriores, a la similitud en requerimientos y funcionalidades con los programas anteriores (excepto el 2) y al nivel de dificultad (uno de los programas más sencillos).

## ¿Cuál fue el % de error en la estimación de tiempo (real/plan-1) en tus programas 1 al 7? En promedio, ¿Cuál fue el % de error en la estimación de tiempo de estos 7 programas juntos?

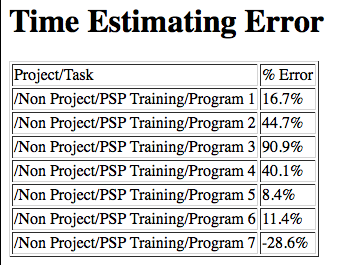


Promedio de error en estimación de tiempo:

* 34.4

La razón por la que tiende a disminuir el % de error en los programas es debido al historial de datos en programas anteriores y la mejora del PSP personal en cuanto a revisiones y estimaciones.

## ¿Cuál fue el programa con mayor error de estimación de tiempo y cuál el menor? ¿Por qué?



Programa con mayor error de estimación de tiempo:

* Programa 3 con 90%

Programa con menor error de estimación de tiempo:

* Programa 5 con 8.4%

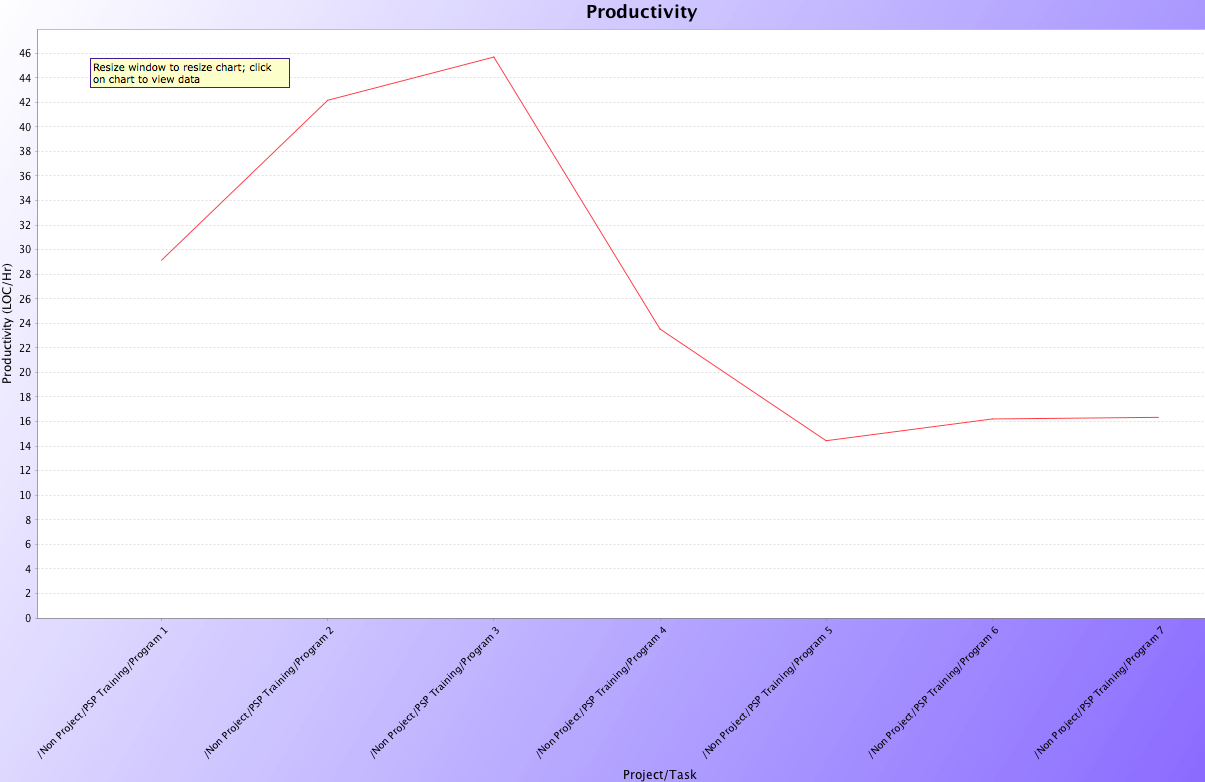
La razón por la que el programa 2 tiene el mayor error de estimación de tiempo se debe a que no se tenían suficientes datos históricos para poder realizar estimaciones, por lo que la estimación se genero con un “juicio personal” sin experiencia.

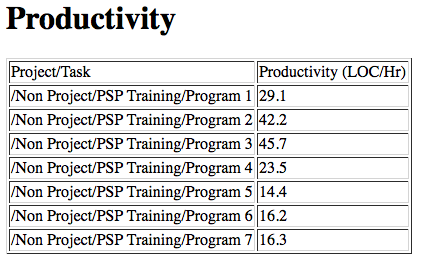
La razón por la que el programa 5 tiene el menor error de estimación de tiempo se debe a que se contaban con datos históricos y experiencia suficiente para realizar estimaciones adecuadas. Además, este programa resulto ser uno de los más sencillos (de hecho fue el de menor tiempo dedicado en horas) por lo que los requerimientos se entendieron de mejor manera y por ello se pudo estimar lo que se necesitaba de manera más eficaz.

## En aquellos programas donde pudiste utilizar el método PROBE “A” o el “B”, ¿El tiempo real estuvo dentro del intervalo de predicción estadístico del 70%? Si no estuvo, ¿Qué pasó?

No se utilizaron los métodos A o B puesto que nunca fueron los más recomendados por Process Dashboard.

## ¿Cuál fue la productividad de cada uno de tus programas? ¿Cuál fue tu productividad promedio?

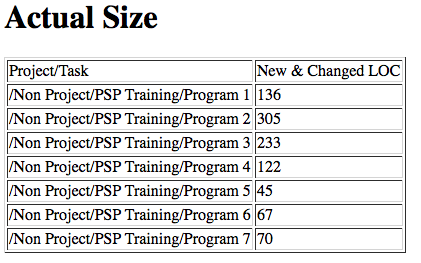




Productividad Promedio:

* 26.771428571

La razón por la que la productividad bajo por programa se debe a que la utilidad de cada LDC mejoro por programa. En otras palabras, existían funciones e instrucciones que se podían mejorar y hasta ignorar en los programas 1 y 2 debido a errores en diseño. Estas LDC no se borraron del código y por ello aumentan la productividad para estos programas, pero en verdad es un incremento falso puesto que estas LDC no se utilizan. De hecho esto se puede ver en el tamaño para cada programa, donde el programa 2 en especifico cuenta con una exagerada diferencia en tamaño comparado con el de los demás.

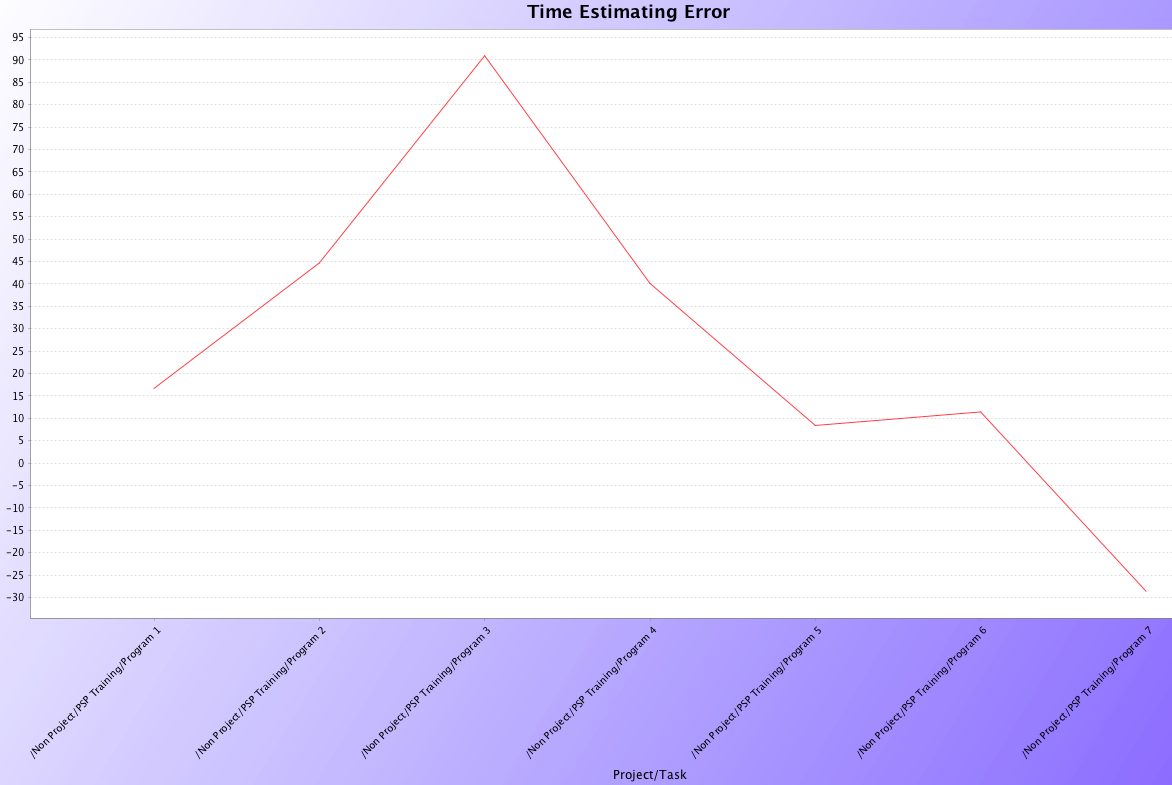


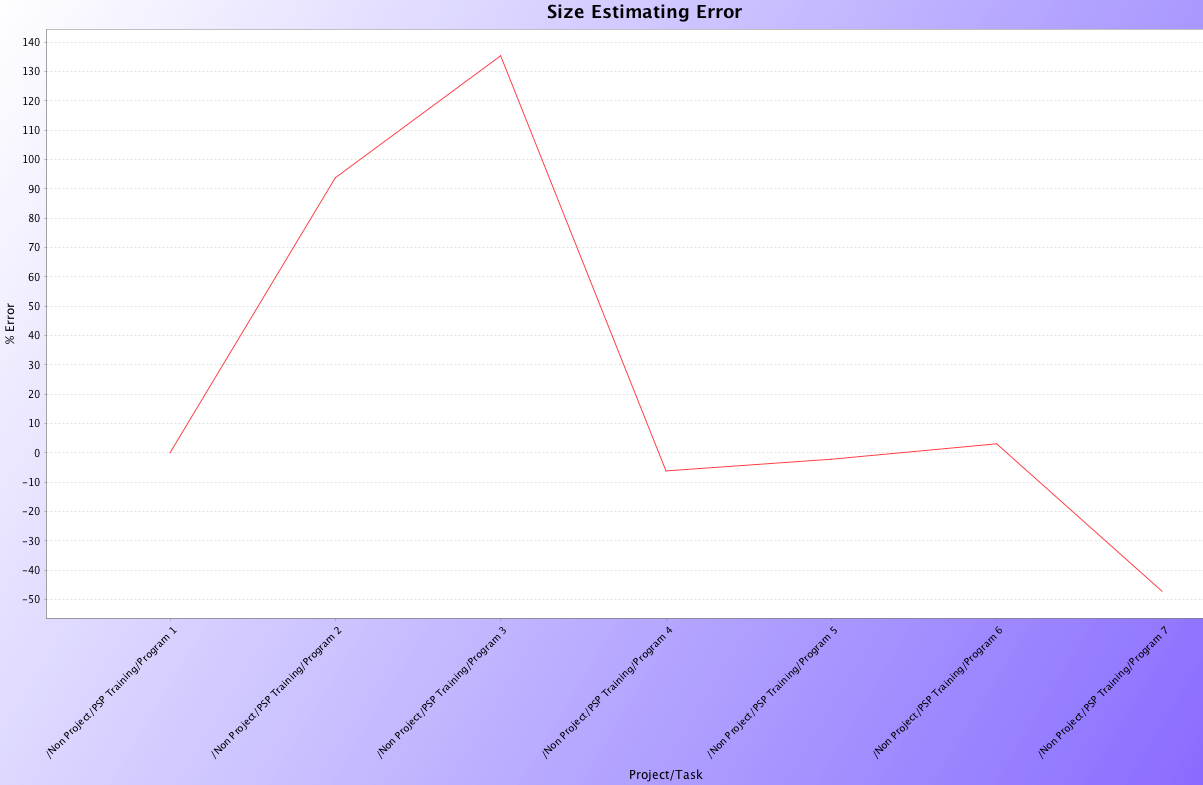
## ¿Cuál es la tendencia de tu productividad? ¿Va a la baja, o al alta? ¿Es inestable? ¿Qué puedes hacer para mejorarla?

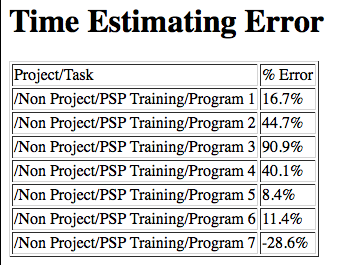
Hay que tomar en cuenta que la productividad para los programas 1 y 2 pudo haber sido inflada de manera accidental. Entonces solo nos quedan los datos para la productivdad en los programas 3 en adelante. Siguiendo estos datos podemos notar que el punto de la gráfica del programa 3 puede ser considerado como un “outlier”, un punto que no se debe utilizar para generar conclusiones por su diferencia relativa con los demás puntos comparables. Entonces solo nos quedna los puntos de los programas 4 en adelante, en donde se puede notar que la tendencia de la productividad es estable alrededor de las 17.4 LDC por hora.

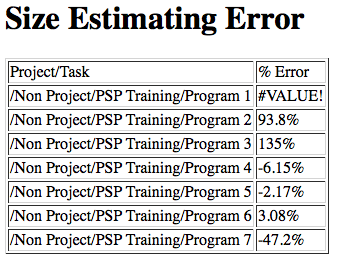
## ¿Qué tanto fueron afectados los errores de estimación de tiempo por tus errores de estimación del tamaño?

De hecho fueron afectados directamente ya que las gráficas para el error de estimación de tiempo y para el error de estimación de tamaó son muy similares. Por lo que se puede concluir que un error en la estimación del tamaño conllevo a un error en la estimación del tiempo, no de la misma magnitud, pero sí en la misma dirección.









## Basado en los datos históricos que has analizado sobre el error de estimación del tiempo, ¿Qué meta realista (alcanzable) te puedes poner para mejorar?

Tener un seguimiento de los porcentajes de error en los programas de mi vida profesional para calcular un porcentaje de error que sirva como “colchon”, el cual sera modificable de acuerdo a programas realizados recientemente y experiencia ganada en la tarea de estimar tamaños. Esto servirá para disminuir el margen de error promedio en los programas futuros para llegar a un margen de error de 20%.

## ¿Qué cambios vas a hacerle a tu proceso personal para alcanzar esta meta?

Calcular de acuerdo a peores escenarios posibles, con el fin de tener un “colchon” y no subestimar al tiempo final del programa considerablemente.

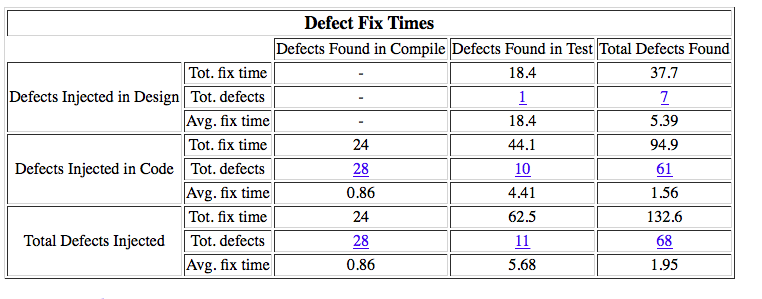
Se debe considerar agregar al tiempo planeado, un porcentaje de margen de error. Este porcentaje debe ser algo apropiado teniendo en cuenta los datos históricos y experiencia personal. Es importante considerar que este margen de error debe ser modificable de acuerdo con la experiencia y mejores predicciones en la estimación de tiempo. Por ejemplo, sería prudente comenzar con un “colchon” de 34.4% para la estimación de tiempo puesto que este fue el promedio obtenido de los programas 2 al 7. No obstante, conforme las predicciones mejoran y el porcentaje de error disminuya, lo mejor debe ser disminuir el “colchon” para que refleje la mejora en la predicción del tiempo de acuerdo a la experiencia.

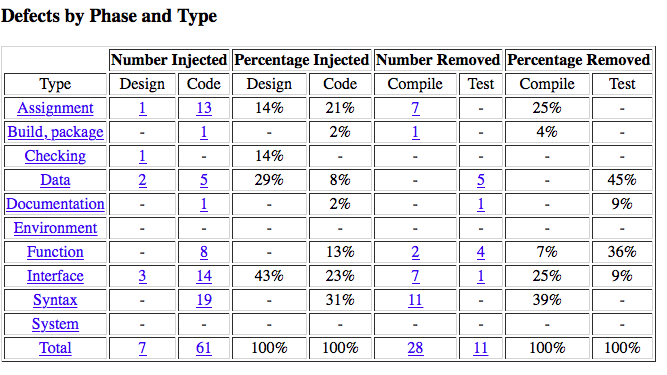
# Análisis de la Calidad (Defectos)

## ¿Qué tipo de defecto es al que más tiempo le dedicaste en la fase de compilación? ¿Y en la fase de pruebas?

Los defectos que tomaron más tiempo en la fase de compilación fueron los inyectados en la fase de código debido a que todos los errores que se encontraton en la fase de compilación pertenecían a la fase de código. Dentro de estos errores inyectados durante la fase de codificación, los errores más frecuentes fueron los de syntax (19), interface (14) y assignment (13), por los que son también los que consumen más tiempo para corregir.

Los defectos que tomaron más tiempo en la fase de pruebas fueron los inyectados en la fase de código debido a que se dedico un total de 44.1 minutos para corregirlos, superior a los 18.4 minutos que se dedico para corregir los errores inyectados en la fase de diseño. Dentro de estos errores inyectados durante la fase de codificación, los errores más frecuentes fueron los de syntax (19), interface (14) y assignment (13), por los que son también los que consumen más tiempo para corregir.





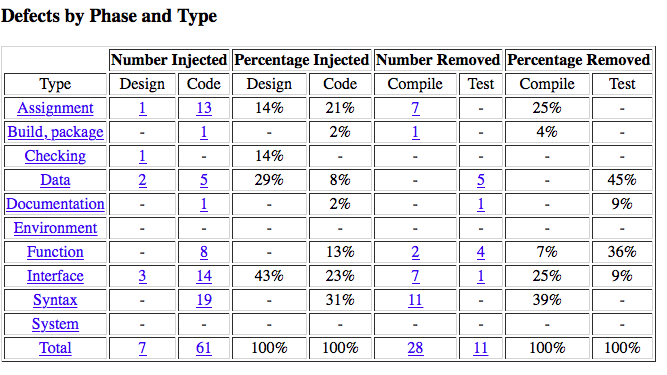
## ¿Qué tipo de defecto es el que más inyectaste en la fase de Diseño? ¿Y en la fase de Codificación?

Error que más se inyecto en la fase de diseño:

* Interface

Error que más se inyecto en la fase de codificación:

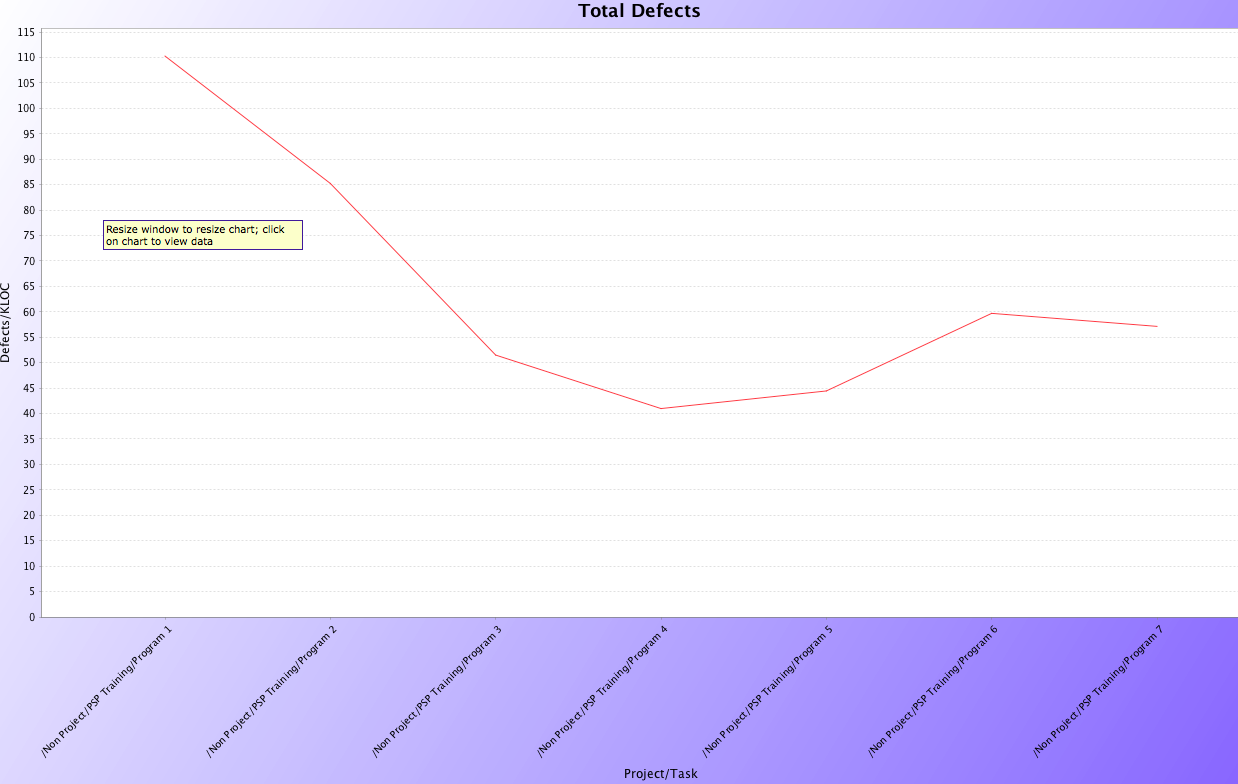
* Syntax



## ¿Cuál es el promedio de horas por defecto dedicadas al remover defectos en las fases de Revisión del Diseño, Revisión del Código, Compilación y Pruebas?

## ¿Cuál es la tendencia en los 7 programas del total de defectos inyectados por KLDC?

La tendencia es a la baja puesto que se implementaron pasos para revisión de analísis y código a partir del programa 3.

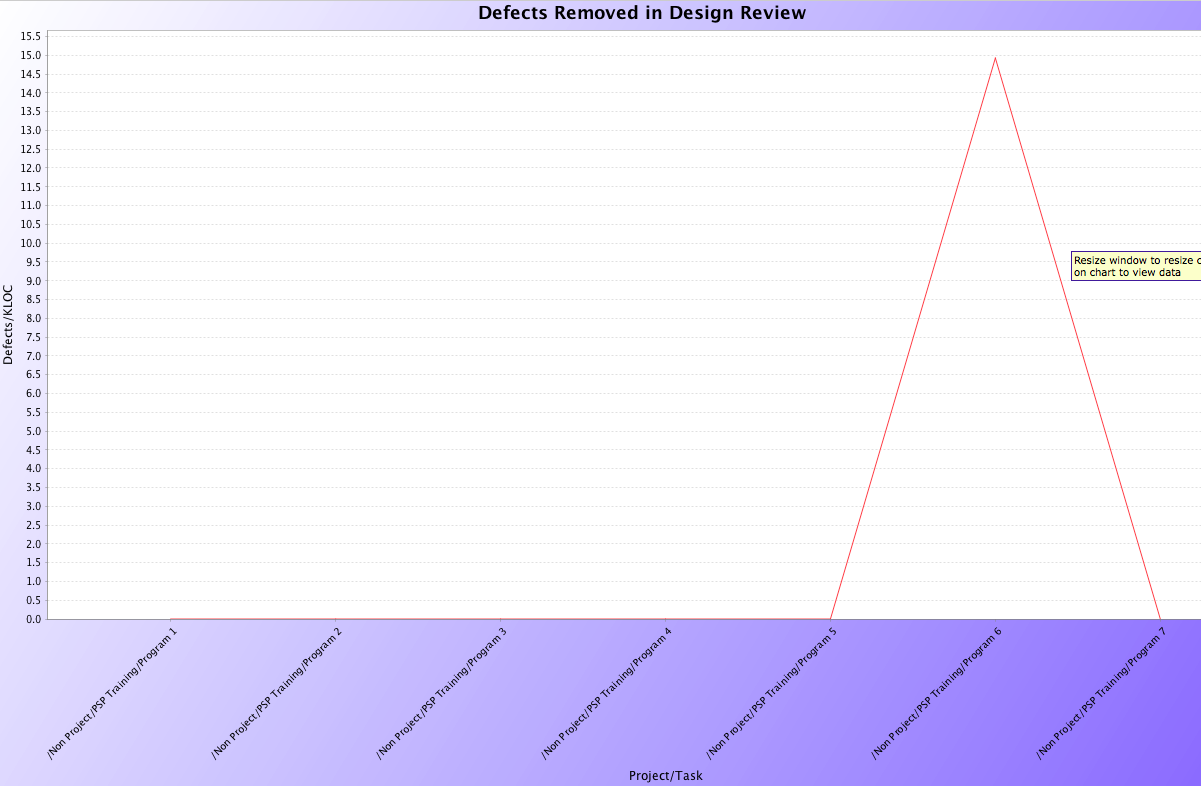


## ¿Cuál es la tendencia en los 7 programas de los defectos removidos por KLDC en las fases de Revisión del Diseño, Revisión del Código, Compilación y Pruebas?

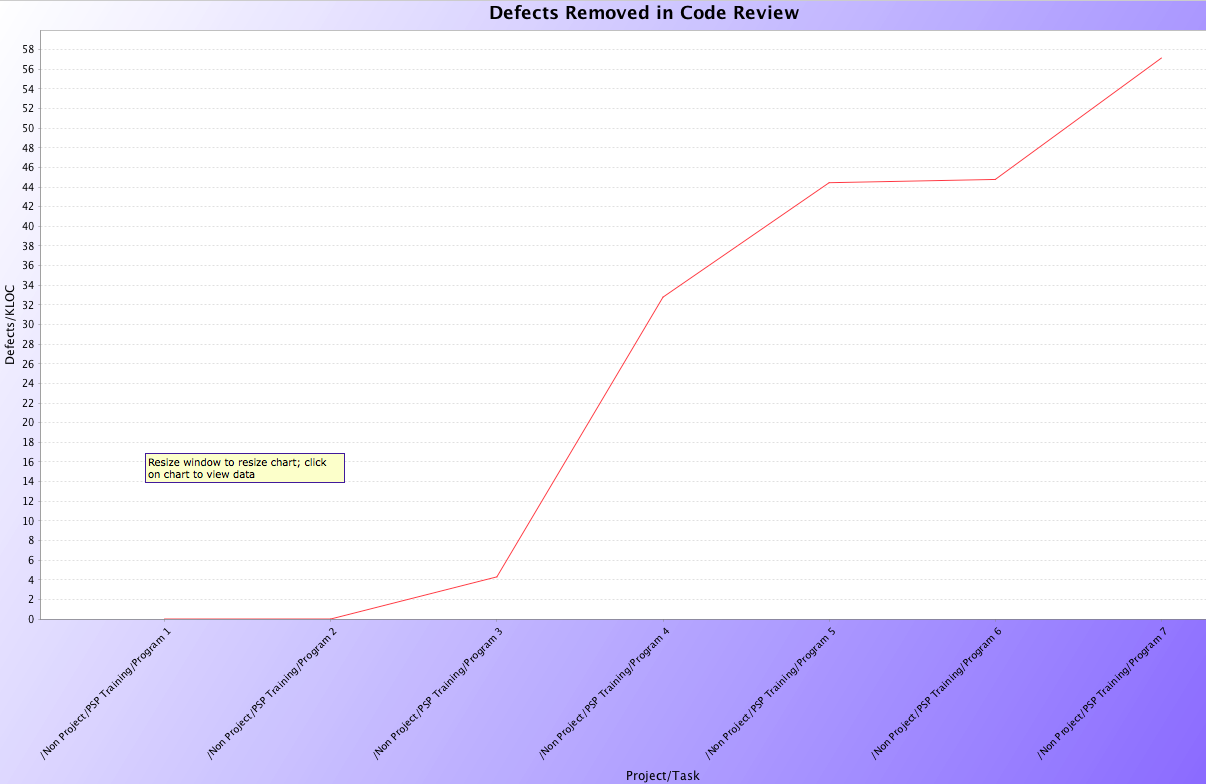
La tendencia de los defectos removidos en la fase de compilación y pruebas es a la baja dado que la tendencía de defectos removidos en la fase de revisión de código es a la alta. Esto ocasiona que menos defectos lleguen a las etapas de compilación y pruebas.

No se puede concluir nada acerca de los defectos removidos en la etapa de revisión de diseño, puesto que solo se cuenta con un dato, por lo que se puede concluir que estos defectos removidos son esporadicos.

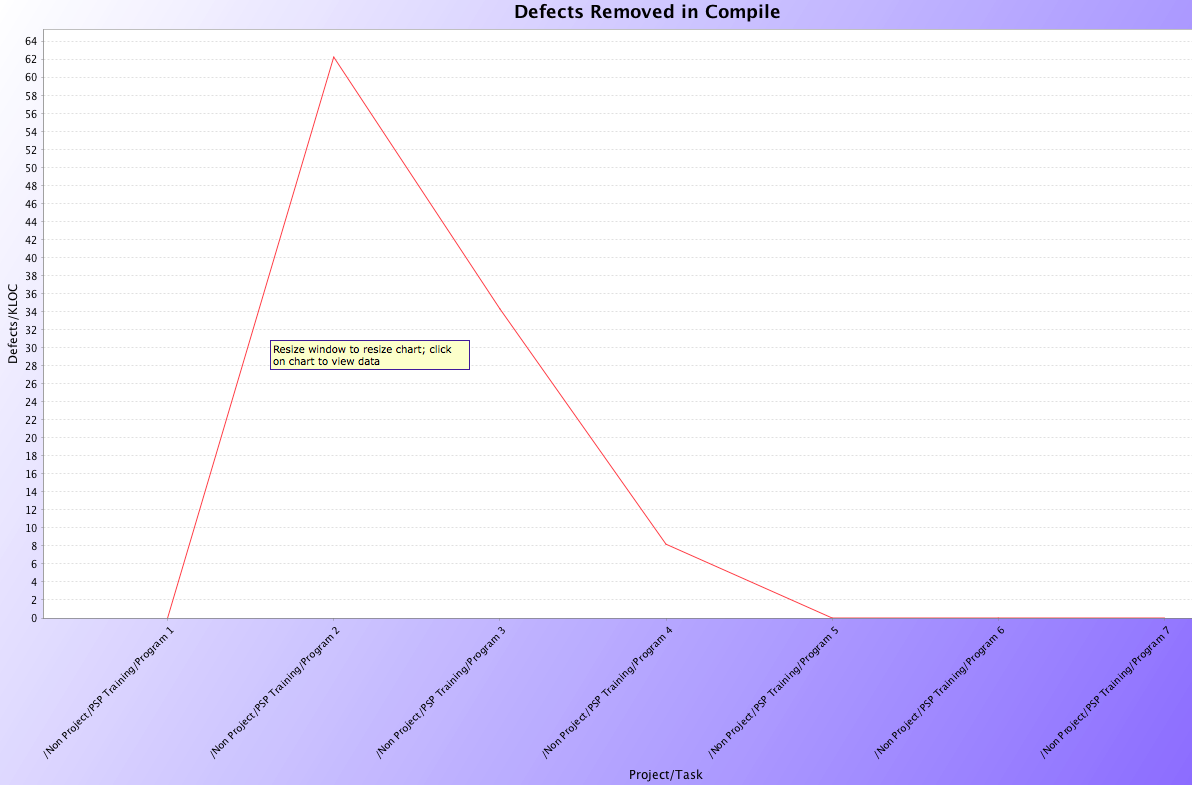
Defectos removidos en la fase de revisión de diseño:



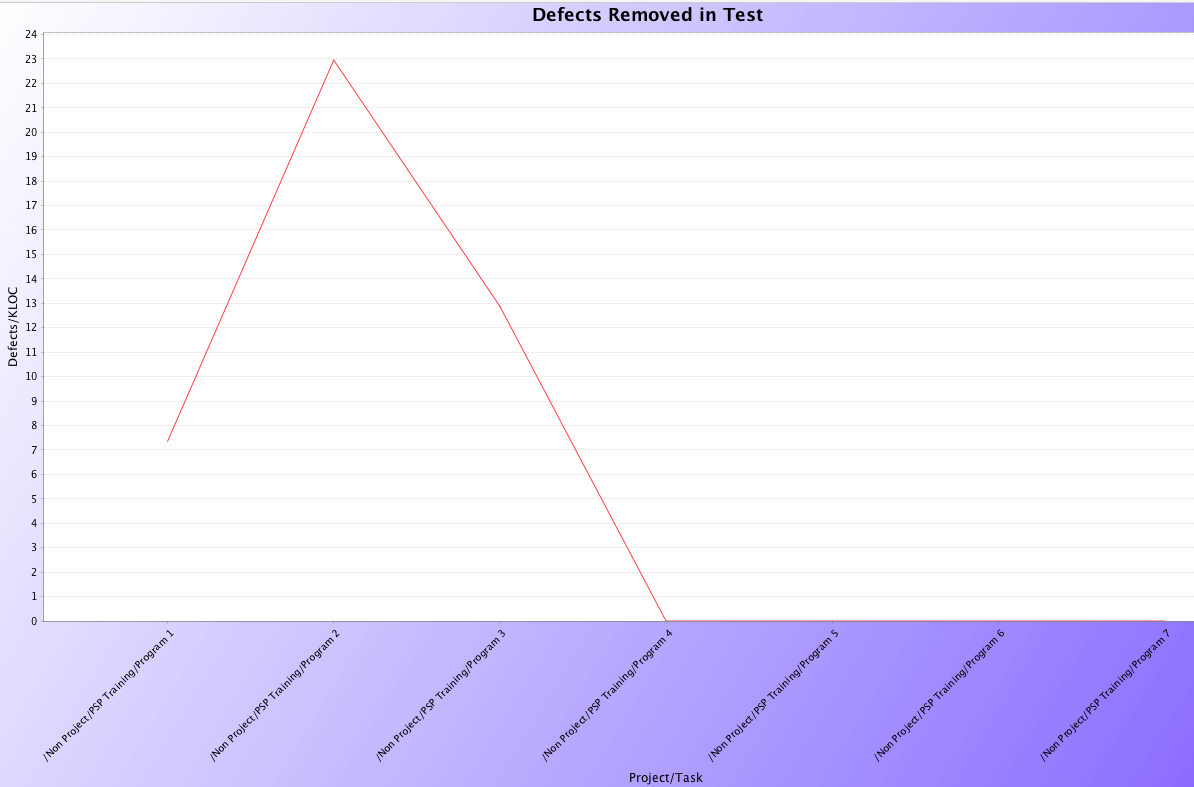
Defectos removidos en la fase de revisión de código:



Defectos removidos en la fase de compilación:

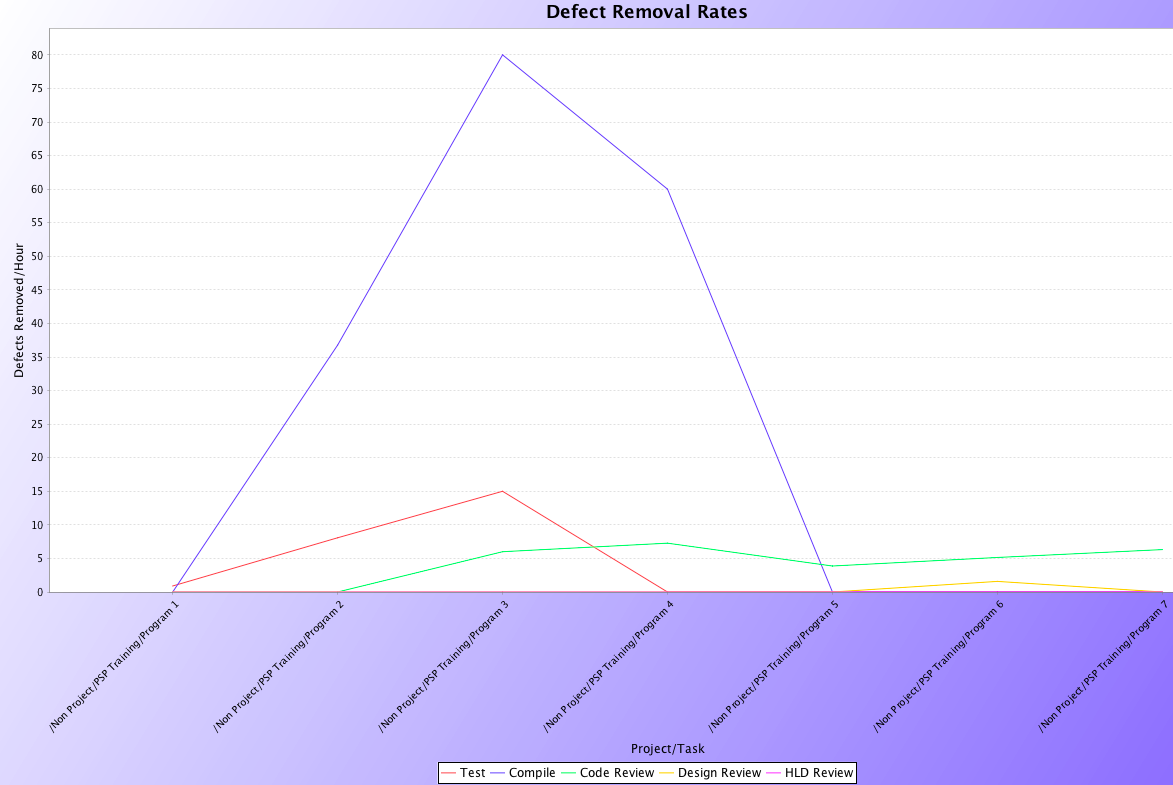


Defectos removidos en la fase de pruebas:



## ¿Cómo se comparan las tasas de remoción de defectos (defectos removidos por hora) en las fases de Revisión del Diseño, Revisión del Código, Compilación y Pruebas en los 7 programas?

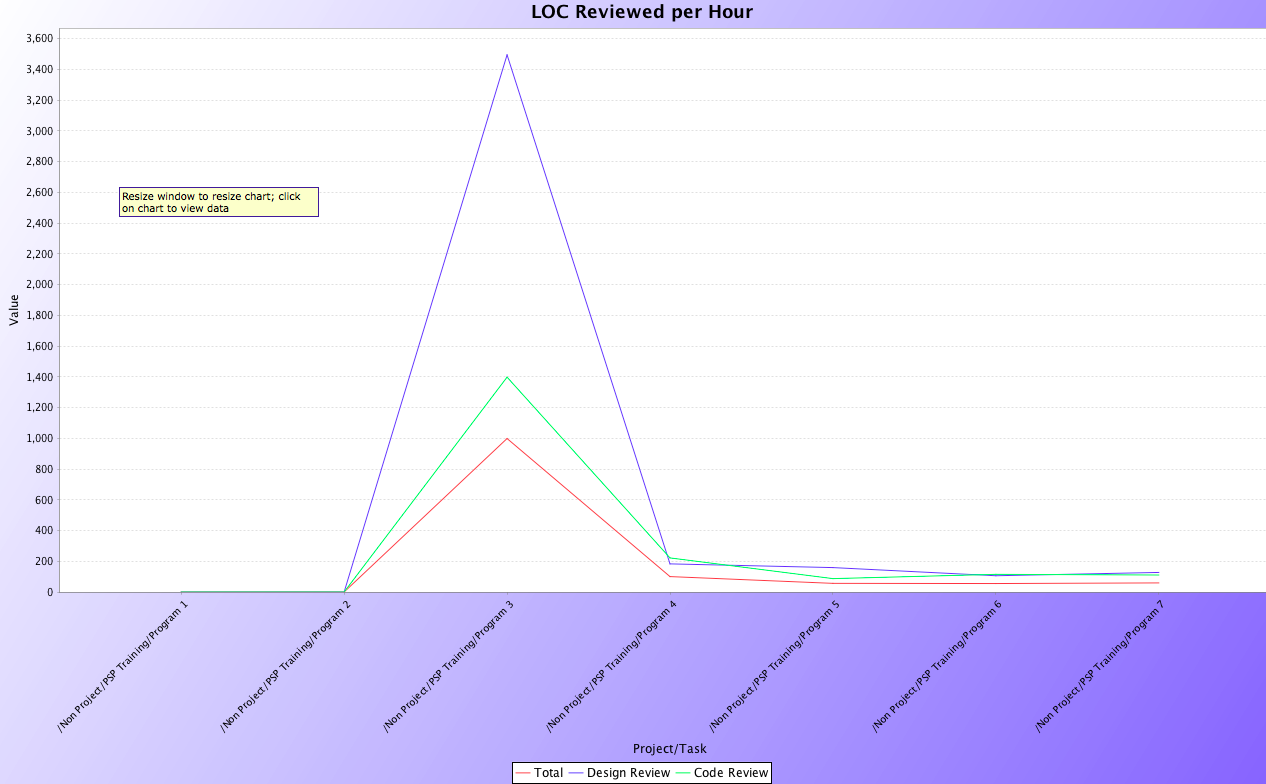
Se puede apreciar que la tasa de remoción de defectos en las fases de compilación y pruebas estaban incrementando hasta el punto en donde la tasa de remoción de defectos para la fase de revisión de diseño y revisión de pruebas comenzo a incrementar, despues de eso lo único que hicieron las tasas de remoción de defctos para las fases de compilación y pruebas fue disminuir.

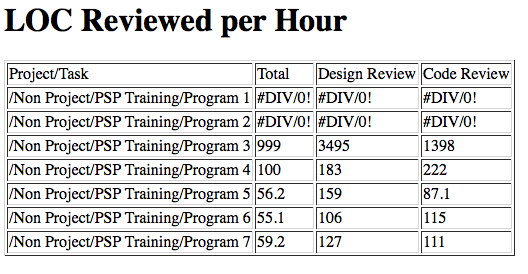


## ¿Cuál fue tu velocidad (LDC por hora) a la que revisaste el código de cada programa?

Es importante tomar en cuenta que los datos para los programas 3 y 4 están equivocados por errores en la revisión. Dicho esto, se deben excluir puntos en la gráfica, lo que haría que su forma fuera más normal.

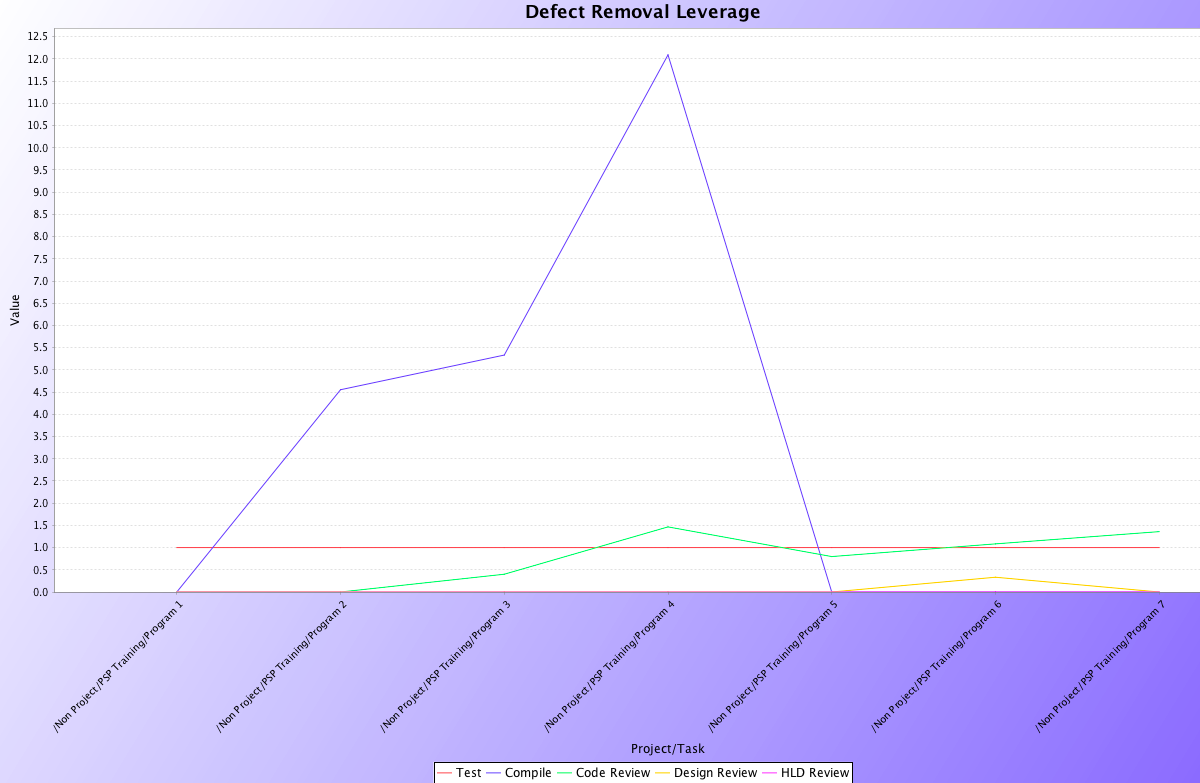
Tomando en cuenta los datos para los programas 5 en adelante se puede concluir que la velocidad promedio es de 104.36666667 LDC por hora.





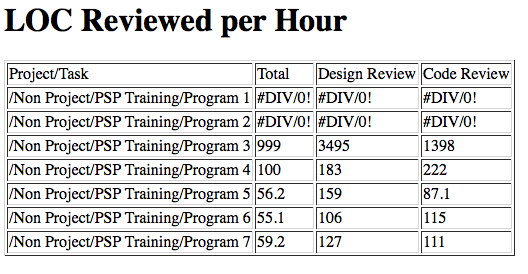
## ¿Cuál fue la palanca de remoción de defectos (DRL) de cada programa para las fases de Revisión del Diseño y Revisión del Código? ¿Es bueno o malo?

Es bueno ya que está incrementando, en especial para la fase de revisión de código, lo que significa que más errores se están corrigiendo en esta fase y evitando que lleguen a la fase de pruebas lo que mejora el yield.



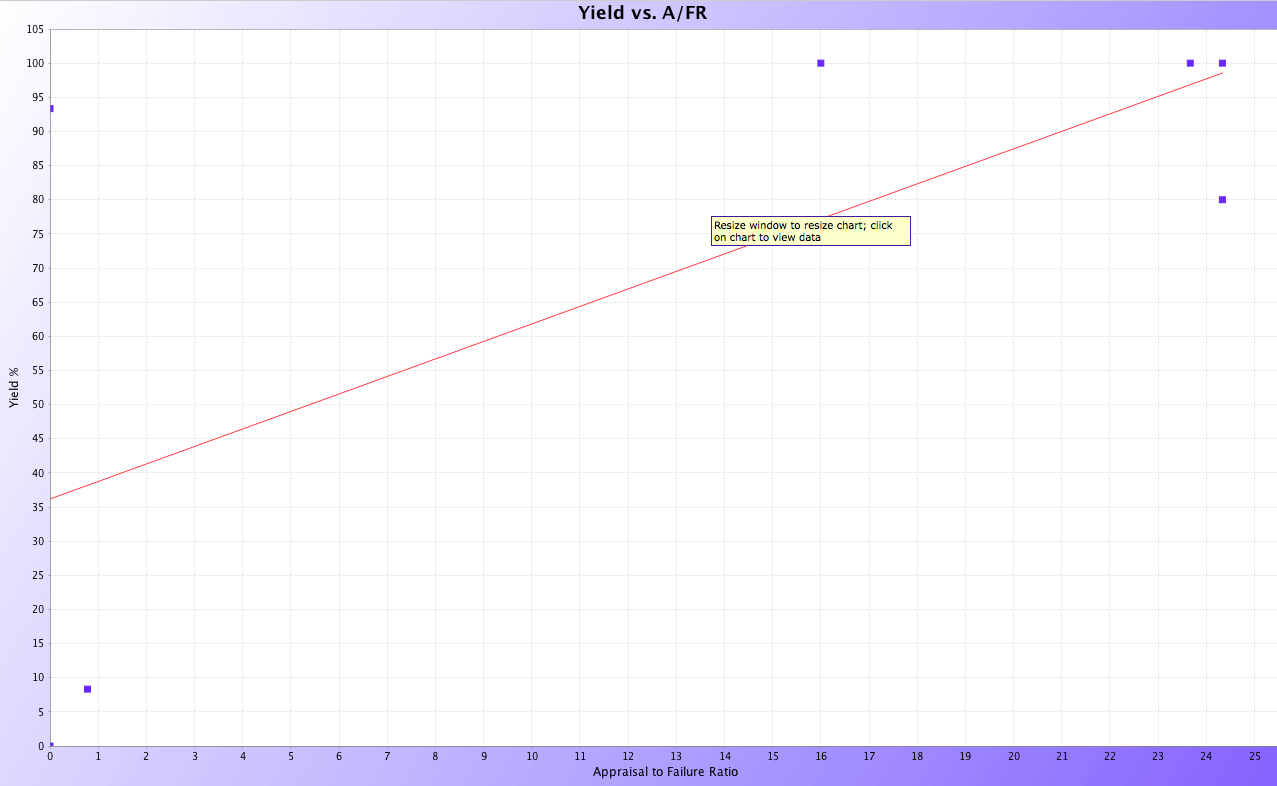
## ¿Existe alguna relación entre el Yield de cada programa y la velocidad (LDC por hora) a la que revisaste el código?

Tomando en cuenta los datos que son comparables de los programas 5 en adelante, no se muestra una relación con el yield.



## ¿Existe alguna relación entre el Yield de cada programa y su A/FR?

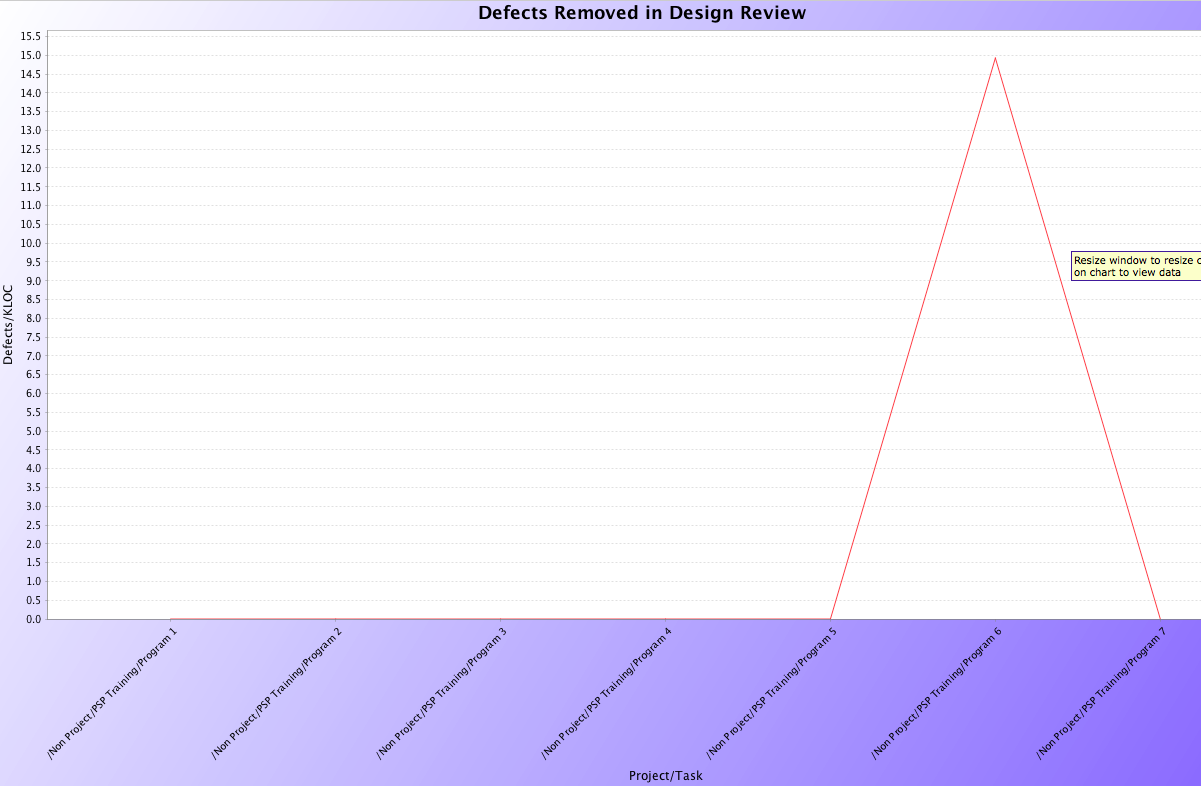
Existe una relación directamente proporcional. Esto tiene que ver con el hecho de que se estaba sacando provecho de las etapas de revisión de código y revisión de diseño para corregir defectos, por lo que el yield de los programas estaba siendo afectado positiviamente.



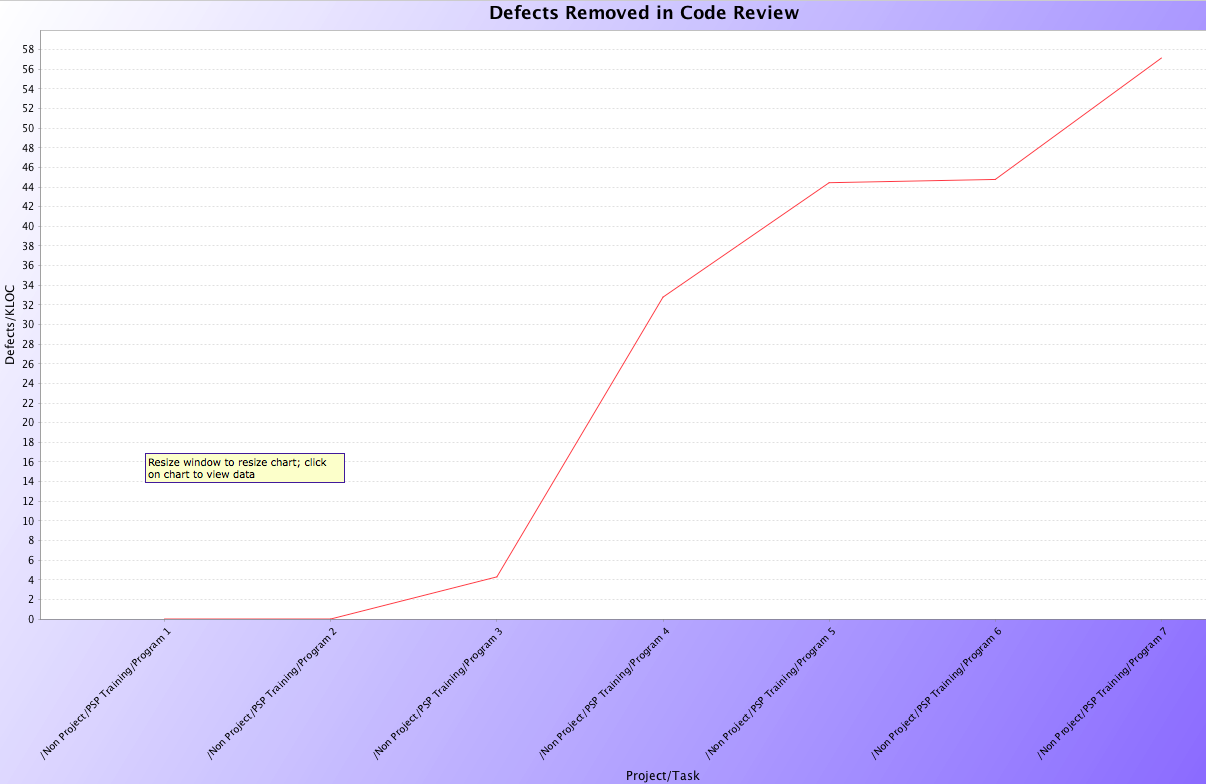
## ¿Estás encontrando los defectos en las fases de Revisión del Diseño y del Código, o se te están escapando a las fases de Compilación y Pruebas? ¿Por qué?

Los defectos se están encontrando en las fases de revisión de diseño y revisión de código puesto que la tendencia para los defectos removidos en la fase de compilación y de pruebas va a la baja, mientras que la tendencia de defectos removidos en la fase de revisión de código está a la alta.

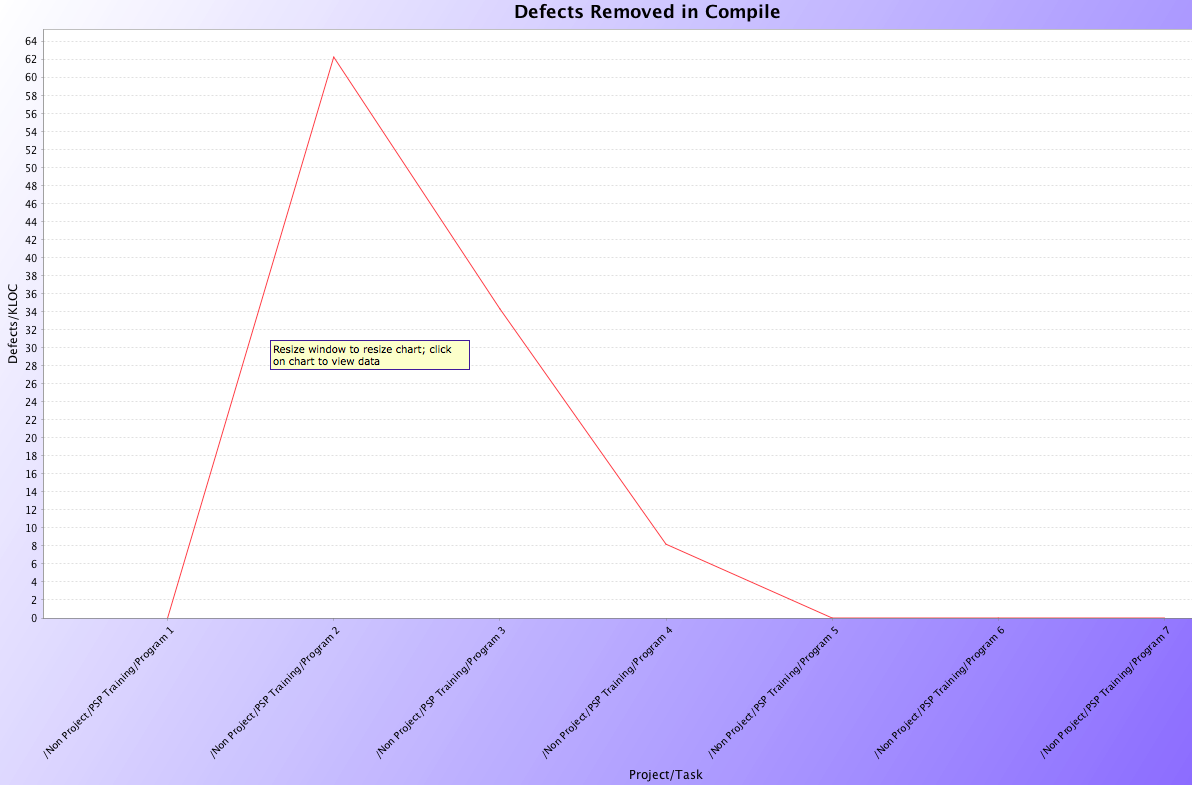
Defectos removidos en la fase de revisión de diseño:



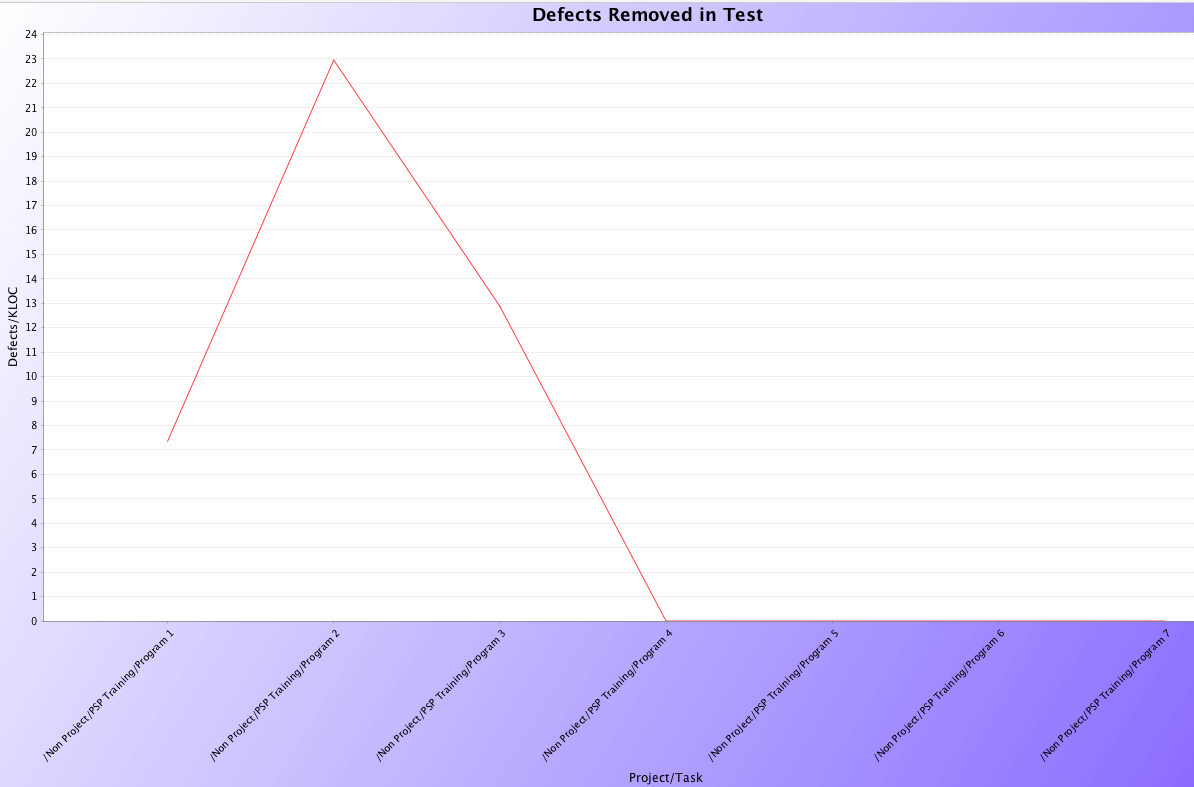
Defectos removidos en la fase de revisión de código:



Defectos removidos en la fase de compilación:



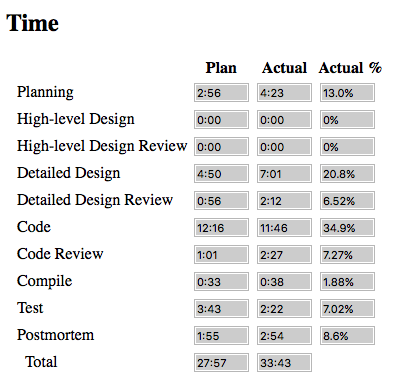
Defectos removidos en la fase de pruebas:



## Basado en los datos históricos que has analizado sobre la calidad de tus programas, ¿Qué meta realista (alcanzable) te puedes poner para mejorar?

Lo ideal sería dedicar más tiempo en las fases de revisión de diseño y revisión de código para lograr que menos errores lleguen a las fases de compilación y pruebas. La meta sería dedicar a las fases de revisión de diseño y revisión de código como mínimo la mitad del tiempo dedicado a las fases de diseño y codificación respectivamente.

Es evidente que el tiempo dedicado a la fase de revisión de código (7.27%) no es ni siquiera la mitad del tiempo dedicado a la fase de codificación (34.9%). De igual manera, el tiempo dedicado a la fase de revisión de diseño (6.52%) no es ni siquiera la mitad del tiempo dedicado a la fase de diseño (20.8%).



## ¿Qué cambios vas a hacerle a tu proceso personal para alcanzar esta meta?

Como soy alguien que aprende y entiende las cosas de manera visual y no tanto de manera lógica, para la fase de revisión de diseño se debe realizar dos diagramas de flujo. El primero debe ser general, simplemente mostrando como se van a comportar y comunicar las clases entre sí. El segundo diagrama de flujo tiene que basarse en la comunicación entre partes del primer diagrama y debe ser especifico, ahora se debe mostrar como los “ítems” se comunican entre si, como se manda y recibe la información, como se realizan los cálculos, etc.

Para la fase de revisión de código se necesita guiarse de manera más estricta en el “checklist para la revisión del código” y utilizar más comentarios estilo “pseudocódigo” como ayudas para leer el código de manera más sencilla, rápida y entendible.

# Conclusión:

Usar el PSP resulta de utilidad para entender los habitos y vicios al momento de programar, con la intención de corregir los defectos antes de que lleguen a las etapas de compilación y pruebas, donde corregirlos resulta costoso en cuanto a tiempo. Personalmente me agrado la introducción al nuevo paradigma de dedicar mas tiempo a las fases de diseño y planeación para no estar diseñando y programando en la fase de codificación. Para llegar a beneficiarse de este paradigma el PSP pone claro que se deben utilizar metricas para generar conclusiones y poder mejorar en el proceso personal. De igual manera, se enfatiza en la importancia de realizar revisiones tanto en diseño y código. Para concluir, con la elaboración de este reporte pude aprender que soy una persona que necesita de ayudas visuales para realizar diseños, que comete muchos errores de tipo de sintaxis al momento de codificar y le benefician mucho las etapas de revisiones para corregir encontrar defectos inyectados, entre otras cosas. Por lo que para seguir mejorando como desarrollador se debe continuar mejorando mi proceso personal de PSP conforme realice nuevos programas en el futuro.

# Párrafo de honestidad:

Por medio de este párrafo yo Hernán Iruegas Villarreal certifico que soy el autor intelectual del material que arriba entrego y que no realicé una copia de ninguna otra persona. En aquellos casos en los que tuve que incluir material de otra persona, reporté su apropiada referencia.