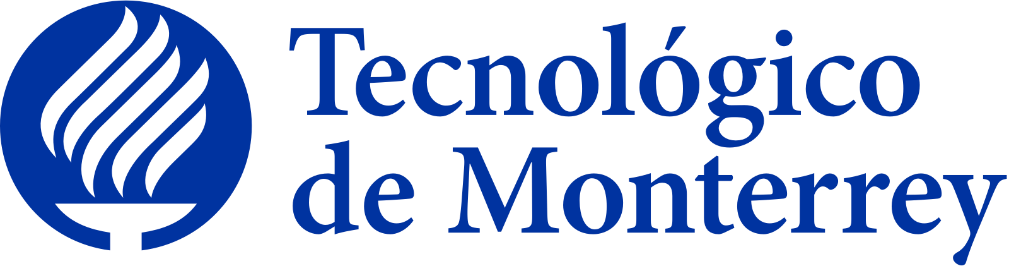
****

**Reporte Final**

**Calidad y pruebas de software**

Lucía Cantú-Miller A01194199

**Profesora:** Adriana González Ugalde

**Instituto Tecnológico y de Estudios Superiores de Monterrey**

**Monterrey, N. L. 19 de noviembre de 2020**.

Índice

[**Introducción** 4](#_Toc56258119)

[**Análisis** **de** **la** **Estimación** **de** **Tamaño** 4](#_Toc56258120)

[1. ¿Cuál fue el tamaño (LDC AyM) de cada uno de tus programas? ¿Cuál fue tu promedio de LDC AyM desarrolladas por programa? 4](#_Toc56258121)

[2. ¿Cuál fue el programa con más LDC AyM y cuál el que tuvo menos? ¿Por qué? 4](#_Toc56258122)

[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](#_Toc56258123)

[4. ¿Cuál fue el programa con mayor error de estimación de tamaño y cuál el menor? ¿Por qué? 5](#_Toc56258124)

[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? 5](#_Toc56258125)

[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](#_Toc56258126)

[7. ¿Qué cambios vas a hacerle a tu proceso personal para alcanzar esta meta? 6](#_Toc56258127)

[**Análisis de la Estimación de Tiempo** 7](#_Toc56258128)

[1. ¿Cuál fue el tiempo en horas dedicado a cada uno de tus programas? ¿Cuál fue tu promedio de horas dedicadas por programa? 7](#_Toc56258129)

[2. ¿Cuál fue el programa al que le dedicaste más tiempo y a cuál menos? ¿Por qué? 7](#_Toc56258130)

[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? 7](#_Toc56258131)

[4. ¿Cuál fue el programa con mayor error de estimación de tiempo y cuál el menor? ¿Por qué? 8](#_Toc56258132)

[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? 8](#_Toc56258133)

[6. ¿Cuál fue la productividad de cada uno de tus programas? ¿Cuál fue tu productividad promedio? 9](#_Toc56258134)

[7. ¿Cuál es la tendencia de tu productividad? ¿Va a la baja, o al alta? ¿Es inestable? ¿Qué puedes hacer para mejorarla? 9](#_Toc56258135)

[8. ¿Qué tanto fueron afectados los errores de estimación de tiempo por tus errores de estimación del tamaño? 9](#_Toc56258136)

[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? 10](#_Toc56258137)

[10. ¿Qué cambios vas a hacerle a tu proceso personal para alcanzar esta meta? 10](#_Toc56258138)

[**Análisis de la Calidad (Defectos)** 10](#_Toc56258139)

[1. ¿Qué tipo de defecto es al que más tiempo le dedicaste en la fase de compilación? ¿Y en la fase de pruebas? 10](#_Toc56258140)

[2. ¿Qué tipo de defecto es el que más inyectaste en la fase de Diseño? ¿Y en la fase de Codificación? 11](#_Toc56258141)

[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? 11](#_Toc56258142)

[4. ¿Cuál es la tendencia en los 7 programas del total de defectos inyectados por KLDC? 12](#_Toc56258143)

[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? 12](#_Toc56258144)

[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? 13](#_Toc56258145)

[7. ¿Cuál fue tu velocidad (LDC por hora) a la que revisaste el código de cada programa? 14](#_Toc56258146)

[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? 14](#_Toc56258147)

[9. ¿Existe alguna relación entre el Yield de cada programa y la velocidad (LDC por hora) a la que revisaste el código? 15](#_Toc56258148)

[10. ¿Existe alguna relación entre el Yield de cada programa y su A/FR? 15](#_Toc56258149)

[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? 16](#_Toc56258150)

[12. Basado en los datos históricos que has analizado sobre la calidad de tus programas, ¿Qué meta realista (alcanzable) te puedes poner para mejorar? 16](#_Toc56258151)

[13. ¿Qué cambios vas a hacerle a tu proceso personal para alcanzar esta meta? 16](#_Toc56258152)

[**Conclusión** 16](#_Toc56258153)

[**Párrafo de honestidad** 17](#_Toc56258154)

# Introducción

Durante este semestre de Agosto – Diciembre 2020 estuvimos realizando 7 programas distintos con la metodología PSP en el laboratorio de Calidad y Pruebas de Software. El propósito de dicho laboratorio es trabajar en nuestras habilidades como programadores y mejorar nuestro proceso de desarrollo de software. Para lograr esto, estuvimos recolectando datos utilizando una herramienta llamada “Process Dashboard”. Con esta herramienta pudimos recolectar el tiempo dedicado en cada fase del desarrollo de cada uno de los programas, los diferentes tipos de defectos inyectados y los removidos.

En este reporte se mostrarán las métricas que fui recolectando en cada uno de los 7 programas que realicé este semestre. Junto con estas métricas llevaré a cabo un análisis de mis resultados en cada una de las distintas áreas (tamaño, tiempo y calidad).

# 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?

|  |  |
| --- | --- |
| **Programa** | **LDC AyM** |
| Programa 1 | 74 |
| Programa 2 | 194 |
| Programa 3 | 100 |
| Programa 4 | 77 |
| Programa 5 | 19 |
| Programa 6 | 62 |
| Programa 7 | 114 |
| **Promedio** | 91.43 |

Analizando los datos recolectados podemos notar que los primeros programas son los que tienen más líneas de código agregadas y modificadas. Esto es debido a que no existía mucho código que se pudiera reutilizar. Luego podemos ver como se reutiliza más código y son menos las líneas agregadas y modificadas.

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

El programa con más LDC AyM es el programa 2 mientras que el programa 5 es el que tuvo menos. Esto es debido a que en el programa 5 tenía más código que podía reusar o utilizar como base. Al reusar código de programas anteriores la cantidad de líneas añadidas y modificadas se reducen. Por otra parte, en el programa 2 no había mucho código que se pudiera reusar y por eso la cantidad de LDC AyM es mayor.

## ¿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?

|  |  |
| --- | --- |
| **Programa** | **% de error en la estimación de tamaño** |
| Programa 2 | 0.71 |
| Programa 3 | -0.23 |
| Programa 4 | -0.27 |
| Programa 5 | 0.05 |
| Programa 6 | 0.29 |
| Programa 7 | 1.05 |
| **Promedio** | 0.266666667 |

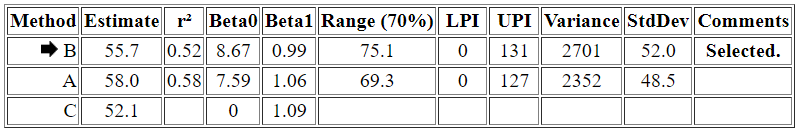
Analizando los datos podemos ver como en los programas 3 y 4 el error de estimación es menor mientras que en el programa 1, 6 y 7 los errores incrementan. Esto es debido a la falta de planeación.

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

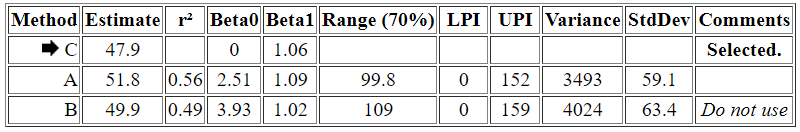
El programa con menor error de estimación fue el programa 5 con un error del 5% mientras que el error de estimación más grande es en el programa 7. Puedo asumir que el programa 5 es el más acertado ya que fue el programa en el que más reutilice código. Por otra parten, en el programa 7 tuve que agregar más líneas de código de lo que planeaba. Esto causando que tenga un error de estimación grande.

## 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?

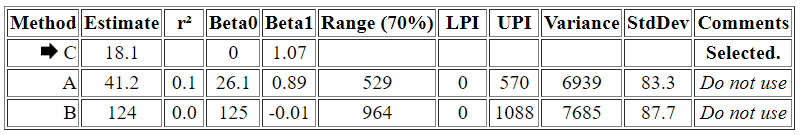
Programa 7



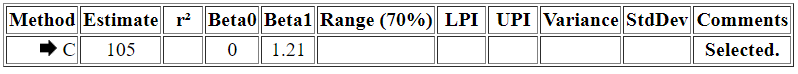
Programa 6



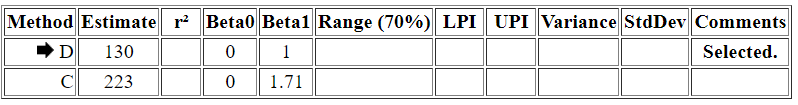
Programa 5



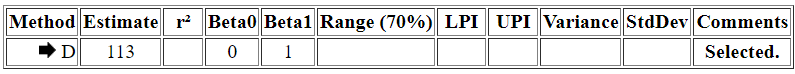
Programa 4



Programa 3



Programa 2



Solo en el programa 7 utilicé el método B para hacer la estimación con un porcentaje de 75.1%. Como expliqué en la pregunta anterior, en este programa no hice una buena estimación del tamaño del programa. Llegue a la conclusión de que al momento de hacer la planeación del programa 7 no había comprendido al 100% los requerimientos del programa y esto causó que hiciera una mala estimación.

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

Después de haber analizado los datos históricos noté que lo que más me ayudaría a mejorar mi estimación de tamaño es primero tener en claro cuales son los requisitos del programa y posteriormente ver que código existente puedo reutilizar.

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

Lo que yo puedo hacer para alcanzar esto es dedicarle un poco más de tiempo a la fase de planeación del programa para así tener una mejor idea de que es lo que voy a tener que hacer durante las siguientes etapas del desarrollo del programa.

# 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?

|  |  |
| --- | --- |
| **Programa** | **Tiempo** |
| Programa 1 | 95 |
| Programa 2 | 207 |
| Programa 3 | 138 |
| Programa 4 | 156 |
| Programa 5 | 74 |
| Programa 6 | 120 |
| Programa 7 | 98 |
| **Promedio** | 126.86 |

Viendo la grafica podemos notar que mediante avanzamos en los programas el tiempo dedicado se reduce. Esto es debido al reuso de código y a la planeación y diseño de los programas.

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

El programa al que menos le dedique tiempo es al programa 5 ya que ese programa es en el que tuve menos LCD AyM ya que la mayor parte del código fue reutilizado. Por otra parte, el programa 2 fue el que más me tomo tiempo ya que no reutilice código y es el programa con la mayor cantidad de LCD AyM.

## ¿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?

|  |  |
| --- | --- |
| **Programa** | **% de error en la estimación de tiempo** |
| Programa 1 | 0.58 |
| Programa 2 | 0.15 |
| Programa 3 | -0.28 |
| Programa 4 | -0.13 |
| Programa 5 | 1.87 |
| Programa 6 | 0.61 |
| Programa 7 | 0.16 |
| **Promedio** | 0.42286 |

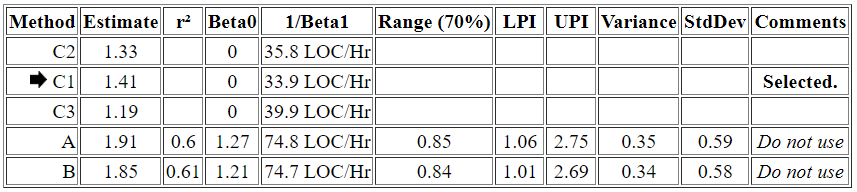
Con estos datos podemos ver que existen muchas variaciones en la estimación de tiempo. Esto es debido a que en ciertos programas no había comprendido del todo las especificaciones del programa. Esto causando que tardara más en el desarrollo del programa.

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

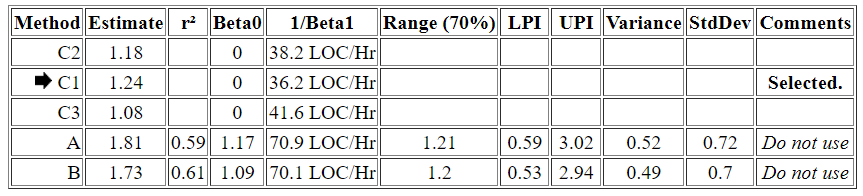
El programa con mayor error de tiempo fue el programa 5, esto debido a que no tome en cuenta de que el tiempo se iba a reducir debido a que una gran parte del código iba a ser reusado de programas anteriores e iban a haber pocas LCD AyM. El programa con menor estimación de tiempo es el programa 4 ya que este era parecido a programas hechos anteriormente y tenia una mejor idea de cuanto me iba a tardar en desarrollarlo.

## 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?

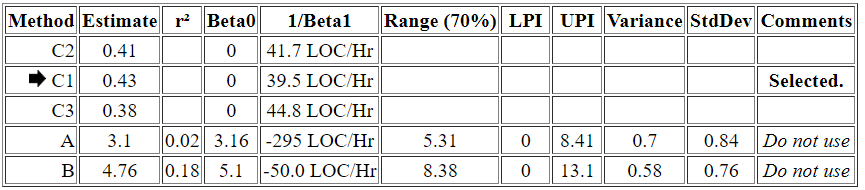
Programa 7



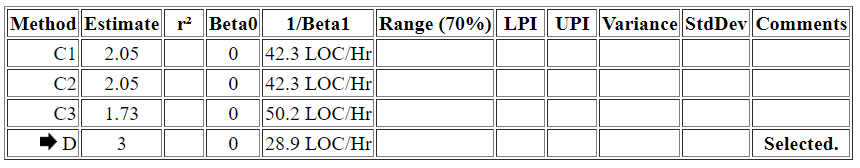
Programa 6



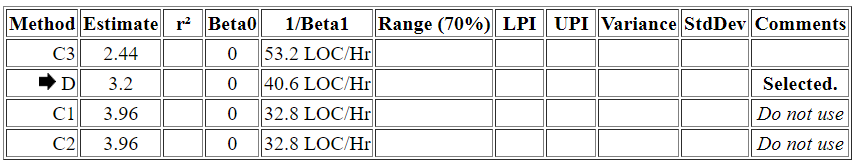
Programa 5



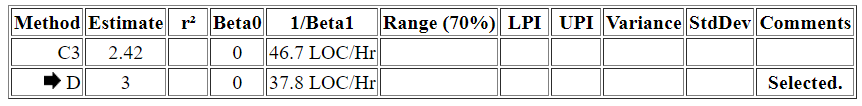
Programa 4



Programa 3



Programa 2



En ninguno de los programas utilice el método A o B.

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

|  |  |
| --- | --- |
| **Programa** | **Productividad** |
| Programa 1 | 46.7 |
| Programa 2 | 56.2 |
| Programa 3 | 43.5 |
| Programa 4 | 29.6 |
| Programa 5 | 15.4 |
| Programa 6 | 31 |
| Programa 7 | 69.8 |
| **Promedio** | 41.74 |

Podemos ver como en todos los programas tuve un nivel de productividad distinto. Esto puede ser efecto de diferentes factores tanto internos como externos al desarrollo del programa.

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

Inicialmente comencé con un buen nivel de productividad, sin embargo, podemos ver como el nivel de productividad fue bajando. Mi productividad se pudo ver afectada por distintos factores externos, así como internos al desarrollo del programa. Lo bueno es que se puede notar como al final logré subir mi productividad. Para mejorar mi productividad puedo dedicarle más tiempo a la fase de planeación y diseño para que al momento de desarrollar el código todo el proceso sea mas fluido.

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

|  |  |  |
| --- | --- | --- |
| **Programa** | **Errores de estimación de tiempo** | **Errores de estimación de tamaño** |
| Programa 1 | 0.58 | 0 |
| Programa 2 | 0.15 | 0.71 |
| Programa 3 | -0.28 | -0.23 |
| Programa 4 | -0.13 | -0.27 |
| Programa 5 | 1.87 | 0.05 |
| Programa 6 | 0.61 | 0.29 |
| Programa 7 | 0.16 | 1.05 |

Se puede notar en la gráfica que el error de estimación de tiempo y la estimación de tamaño son variable. Se puede asumir que los errores de estimación de tamaño si tienen un impacto en los errores de estimación de tiempo.

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

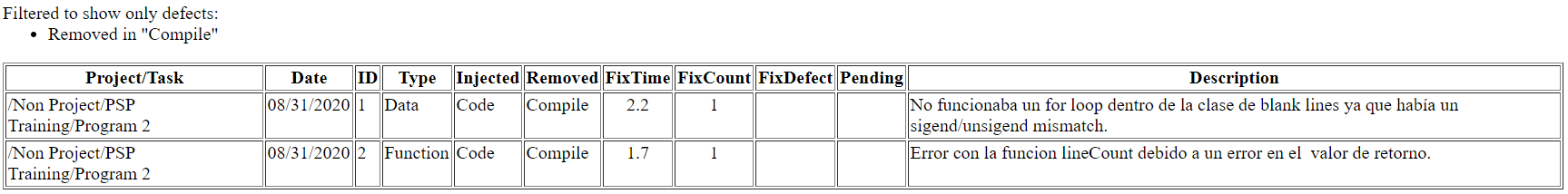
Viendo los resultados de la estimación de tiempo puedo llegar a la conclusión de que la mejor manera de tener estimaciones más acertadas es tener una buena comprensión de programa y de que es lo que se va a tener que hacer durante su desarrollo. La meta es poder tener una mejor comprensión del programa a realizar.

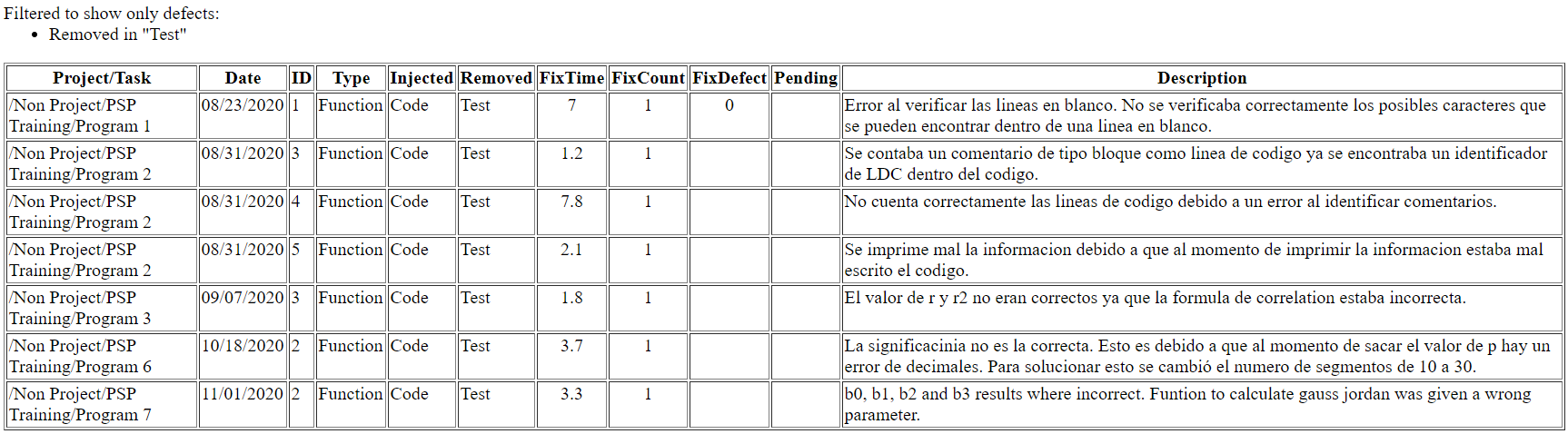
## ¿Qué cambios vas a hacerle a tu proceso personal para alcanzar esta meta?

Como dicho en la pregunta anterior, dedicarle más tiempo a tener a la fase de planeación para asegurarnos de que entendemos bien los requisitos del programa me puede ayudar a mejorar y alcanzar la meta.

# Análisis de la Calidad (Defectos)

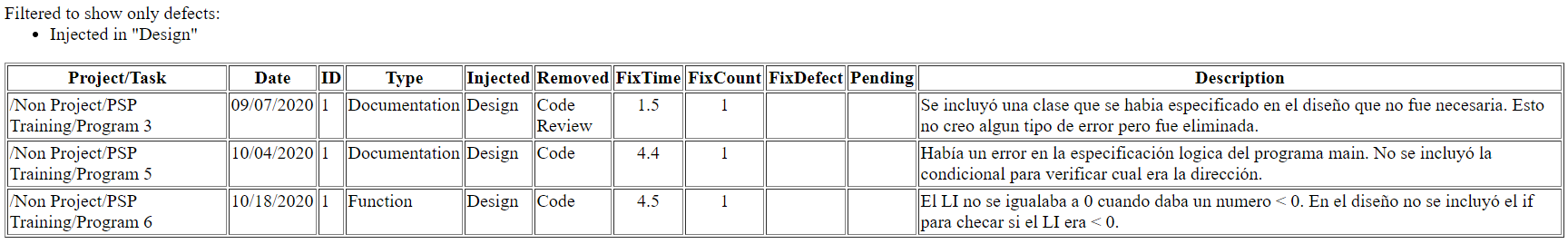
## ¿Qué tipo de defecto es al que más tiempo le dedicaste en la fase de compilación? ¿Y en la fase de pruebas?

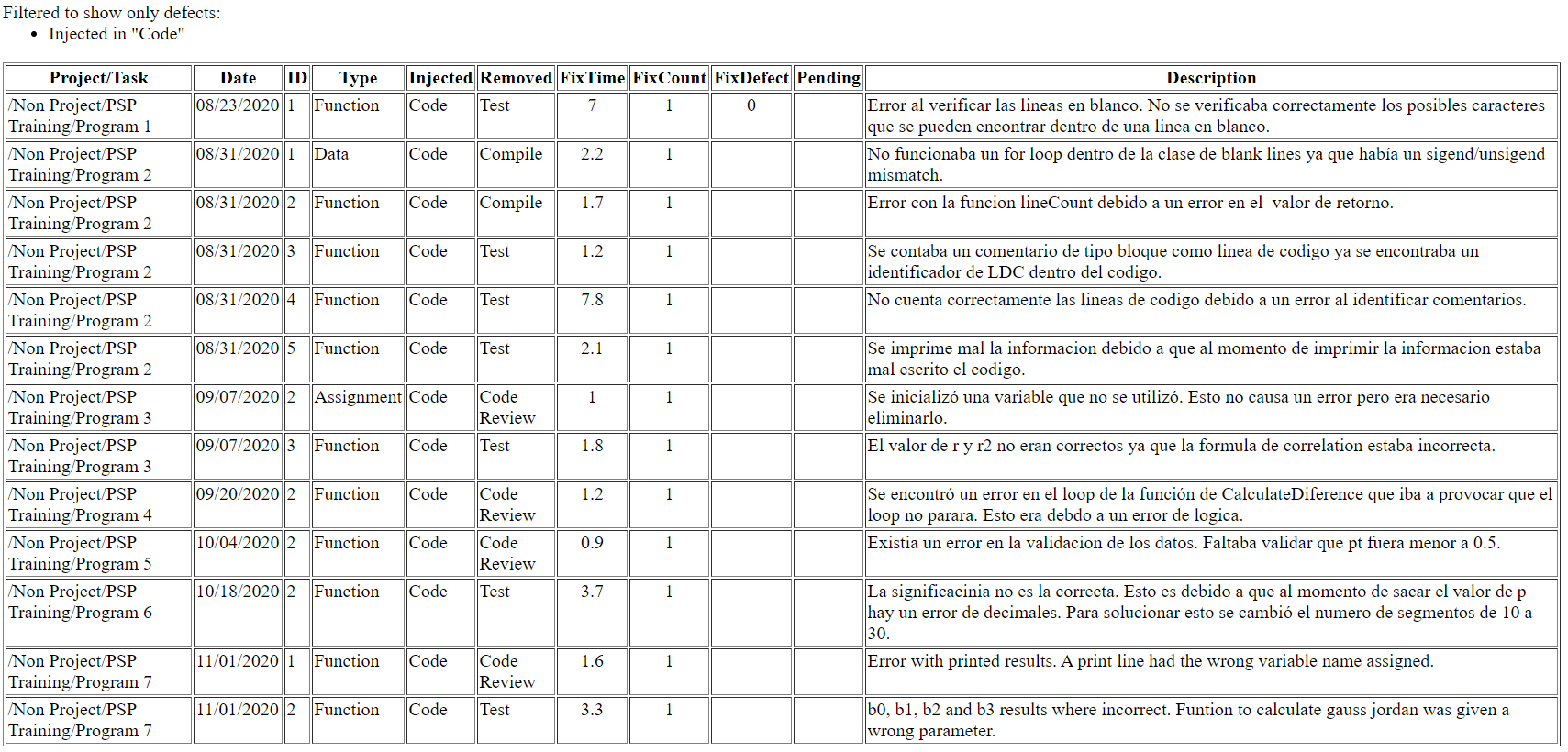




En la fase de compilación solo tuve dos defectos. Uno de “data” y el otro de “function”. En la fase de pruebas todos los defectos fueron de “function”.

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





En la fase de diseño se inyectaron más defectos de documentación. En la fase de codigo la mayoría de los defectos fueron de “function”.

## ¿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?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Programa** | **Test** | **Compile** | **Code Review** | **Design Review** |
| Programa 1 | 5 | 0 | 0 | 0 |
| Programa 2 | 7.5 | 24 | 0 | 0 |
| Programa 3 | 12 | 0 | 17.1 | 0 |
| Programa 4 | 0 | 0 | 15 | 0 |
| Programa 5 | 0 | 0 | 10 | 0 |
| Programa 6 | 8.57 | 0 | 0 | 0 |
| Programa 7 | 15 | 0 | 12 | 0 |
| **Promedio** | 6.87 | 3.43 | 7.73 | 0 |

Analizando los datos podemos ver que la mayor cantidad de tiempo dedicado a los defectos es durante la fase de pruebas y revisión de código. Mientras que muy pocas horas fueron dedicadas en la fase de revisión de diseño y compilación.

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

|  |  |
| --- | --- |
| **Programa** | **Densidad de Defectos** |
| Programa 1 | 13.5 |
| Programa 2 | 25.8 |
| Programa 3 | 30 |
| Programa 4 | 13 |
| Programa 5 | 105 |
| Programa 6 | 32.3 |
| Programa 7 | 17.5 |
| **Promedio** | 33.87 |

Viendo la grafica podemos ver que la cantidad de defectos inyectados es variable pero se tiene puede concluir que el promedio es de 33.87 defectos inyectados por KLDC.

## ¿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?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Programa** | **Revisión del diseño /Defectos Removidos** | **Revisión de código /Defectos Removidos** | **Compilación /Defectos Removidos** | **Pruebas/Defectos Removidos** |
| Programa 1 | 0 | 0 | 0 | 1 |
| Programa 2 | 0 | 0 | 2 | 3 |
| Programa 3 | 0 | 2 | 0 | 1 |
| Programa 4 | 0 | 1 | 0 | 0 |
| Programa 5 | 0 | 1 | 0 | 0 |
| Programa 6 | 0 | 0 | 0 | 1 |
| Programa 7 | 0 | 1 | 0 | 1 |

Analizando la grafica podemos notar que los datos recolectados son muy variables dentro de todas las fases y programas.

## ¿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?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Programa** | **Revisión de diseño/Tasa de Defectos Removidos** | **Revisión de código/Tasa de Defectos Removidos** | **Compilación/Tasa de Defectos Removidos** | **Pruebas/Tasa de Defectos Removidos** |
| Programa 1 | 0 | 0 | 0 | 5 |
| Programa 2 | 0 | 0 | 24 | 7.5 |
| Programa 3 | 0 | 17.1 | 0 | 12 |
| Programa 4 | 0 | 15 | 0 | 0 |
| Programa 5 | 0 | 10 | 0 | 0 |
| Programa 6 | 0 | 0 | 0 | 8.57 |
| Programa 7 | 0 | 12 | 0 | 15 |

Como fue establecido anteriormente, los datos recolectados son muy variables dentro de todas las fases y programas.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Programa** | **Total** | **Design Review** | **Code Review** |
| Programa 1 | 0 | 0 | 0 |
| Programa 2 | 0 | 0 | 0 |
| Programa 3 | 600 | 2000 | 857 |
| Programa 4 | 243 | 308 | 1155 |
| Programa 5 | 87.7 | 163 | 190 |
| Programa 6 | 169 | 286 | 413 |
| Programa 7 | 570 | 977 | 1368 |

Analizando los datos podemos ver que los programas donde más líneas de código fueron agregadas son las que más tomaron tiempo para revisar mientras que los programas donde se reutilizó más código (por ejemplo, el programa 5) se tomó menos tiempo revisar.

## ¿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?

|  |  |  |
| --- | --- | --- |
| **Programa** | **Design Review/DRL** | **Code Review/DRL** |
| Programa 1 | 0 | 0 |
| Programa 2 | 0 | 0 |
| Programa 3 | 0 | 1.43 |
| Programa 4 | 0 | 2.2 |
| Programa 5 | 0 | 1.57 |
| Programa 6 | 0 | 0 |
| Programa 7 | 0 | 0.8 |

Podemos ver que el valor del DRL en la mayor parte de los programas es ≥1.0. Esto es bueno, pero se debe de trabajar en mejorar el DRL ya que si existen proyectos donde es < 1.

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Programa** | **Total** | **Design Review** | **Code Review** | **Yield** |
| Programa 1 | 0 | 0 | 0 | 0 |
| Programa 2 | 0 | 0 | 0 | 0 |
| Programa 3 | 600 | 2000 | 857 | 66.70% |
| Programa 4 | 243 | 308 | 1155 | 100% |
| Programa 5 | 87.7 | 163 | 190 | 100% |
| Programa 6 | 169 | 286 | 413 | 50% |
| Programa 7 | 570 | 977 | 1368 | 50% |

En los programas donde se tiene un Yield de 100% se tienen una velocidad de revisión menor comparada a los programas que se tiene un Yield menor.

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

|  |  |  |
| --- | --- | --- |
| **Programa** | **Yield** | **AFR** |
| Programa 1 | 0% | 0 |
| Programa 2 | 0% | 0 |
| Programa 3 | 66.7% | 2 |
| Programa 4 | 100% | 6.33 |
| Programa 5 | 100% | 4.33 |
| Programa 6 | 50% | 3.14 |
| Programa 7 | 50% | 3 |

En los programas donde se tiene un yield del 100% el A/FR es mayor mientras que en los otros programas donde el yield es menor el A/FR es menor. En la gráfica podemos notar que tienen una tendencia similar.

## ¿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?

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Planning** | **HLD** | **HLD Review** | **Design** | **Design Review** | **Code** | **Code Review** | **Compile** | **Test** | **Reassessment** | **Postmortem** |
| 0 | 0 | 0 | 0 | 0 | 2 | 5 | 2 | 7 | 0 | 0 |

La mayor parte de los defectos los encuentro en la fase de pruebas. Esto es debido a que no le dedico mucho tiempo a las fases de revisión ya que se me hace más fácil encontrar y resolver los errores en la fase de pruebas. Sin embargo, también encuentro una buena cantidad de errores en la fase de revisión de código.

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

Basado en los datos históricos puedo ver que no tengo una cantidad muy grande de errores en cada programa sin embargo estos errores se podrían reducir. La meta es reducir la cantidad de errores encontrados en la fase de pruebas.

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

Para poder lograr esto le podría dedicar más tiempo a verificar que mi código esté bien escrito y que aparte cumpla con los requerimientos. Para poder darme cuenta de que si cumple con los requerimientos le podría dedicar más tiempo a la fase de planeación y diseño.

# Conclusión:

Después de haber hecho un análisis de los datos recolectados a lo largo del semestre puedo concluir que el PSP es un buen método que puedo implementar en el desarrollo de mis proyectos ya que me ayuda a mejorar mi productividad y a tener mejores resultados y menos errores en mis programas. En este semestre me di cuenta de lo importante que nuestros proyectos tengan una excelente calidad ya que esto nos ayuda en un futuro ya que todo lo que hacemos puede ser utilizado en un futuro. El haber trabajado en estos programas y documentar los tiempos, tamaños y errores me ha ayudado a tener una mejor visión sobre lo importante que es la calidad en desarrollo de software.

# Párrafo de honestidad:

Por medio de este párrafo yo Lucía Cantú-Miller 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.