



**Sistema Qsource en la calidad del software desarrollado
en RPG**

**TESIS PARA OPTAR EL GRADO ACADÉMICO DE:
Maestro en Gestión de Tecnologías de Información**

AUTOR:

Br. Giovanni Barrero Ortiz

ASESOR:

Dr. Willian Sebastián Flores Sotelo

SECCIÓN:

Ingeniería

LÍNEA DE INVESTIGACIÓN:

Sistemas Basados en Gestión de Procesos de Negocio

PERÚ – 2018

Página del Jurado

Presidente

Dr. Cesar Humberto Del Castillo Talledo

Secretario

Dr. Willian Sebastián Flores Sotelo

Vocal

Dedicatoria

Este trabajo de investigación se lo dedico a mi madre Ligia Ortiz y a mi padre Rafael Barrero quienes siempre me han apoyado, a mi esposa Virna Lira y mis hijos Paolo Barrero y Camila Barrero quienes alegran mi vida y me dan apoyo constante para el cumplimiento de mis metas y objetivos y a mis hermanos.

Giovanni Barrero Ortiz.

Agradecimiento

Agradezco a la universidad Cesar Vallejo por su aporte a mi vida profesional, También expreso mi sincero agradecimiento al Dr. Willian Sebastián Flores Sotelo, por el importante soporte en el desarrollo de la tesis. De manera especial y sincera a la institución donde laboro al Banco AGROBANCO, por darme la oportunidad de laborar en tan prestigiosa institución, pero a la vez por darme las facilidades para el desarrollo de mi tesis. A Dios y a mis familiares, compañeros de trabajo, amigos y a todas aquellas personas que de una u otra forma contribuyeron con el desarrollo de la tesis. Giovanni Barrero Ortiz.

Declaratoria de autenticidad

Yo, Giovanni Barrero Ortiz, estudiante del Programa de Maestría en Gestión de Tecnologías de Información de la Escuela de Postgrado de la Universidad César Vallejo, identificado con DNI N° 48839223, respectivamente, con la tesis titulada Sistema QSOURCE en la calidad del software desarrollado en RPG, declaro bajo juramento que:

- 1) La tesis es de autoría propia.
- 2) Se ha respetado las normas internacionales de citas y referencias para las fuentes consultadas. Por tanto, la tesis no ha sido plagiada ni total ni parcialmente.
- 3) La tesis no ha sido auto plagiada; es decir, no ha sido publicada ni presentada anteriormente para obtener algún grado académico previo o título profesional.
- 4) Los datos presentados en los resultados son reales, no han sido falseados, ni duplicados, ni copiados y por tanto los resultados que se presenten en la tesis se constituirán en aportes a la realidad investigada.

De identificarse la presencia de fraude (datos falsos), plagio (información sin citar a autores), auto plagio (presentar como nuevo algún trabajo de investigación propio que ya ha sido publicado), piratería (uso ilegal de información ajena) o falsificación (representar falsamente las ideas de otros), asumimos las consecuencias y sanciones que de nuestras acciones se deriven, sometiéndonos a la normatividad vigente de la Universidad César Vallejo.

Los Olivos, marzo 18 del 2018

Giovanni Barrero Ortiz

D.N.I 48839223

Presentación

Señores miembros del jurado calificador

De conformidad con el Reglamento de Grados y Títulos de la Universidad César Vallejo, pongo a vuestra consideración la evaluación de la tesis “Sistema QSOURCE en la calidad del software desarrollado en RPG” elaborada con el objetivo general de describir el efecto de la implementación del sistema QSOURCE en la mejora de la calidad del software desarrollado en lenguaje RPG.

El estudio comprende los siguientes capítulos: el capítulo I se refiere a la introducción; el capítulo II se refiere al Marco metodológico; el capítulo IV se refiere a la discusión; el capítulo V a las conclusiones; el capítulo VI a las recomendaciones. Por último, el capítulo VII menciona las referencias bibliográficas y los anexos respectivos.

Señores miembros del jurado esperamos que esta investigación sea evaluada y merezca su aprobación.

Los Olivos, marzo 18del 2018

Giovanni Barrero Ortiz

Indice de Contenido

Página del Jurado	ii
Dedicatoria	iii
Agradecimiento	iv
Declaratoria de autenticidad	v
Presentación	vi
Resumen	xix
Abstract	xx
I Introducción	1
1.1 Realidad problemática	2
1.2 Trabajos previos	2
1.2.1 Trabajos previos internacionales	2
1.2.2 Trabajos previos nacionales	7
1.3 Teorías relacionadas al tema	9
1.3.1 Sistema QSOURCE	9
1.4 Problema	72
1.4.1 Problema general	72
1.5 Justificación del estudio	73
1.5.1 Justificación teórica	73
1.5.2 Justificación Práctica	73
1.5.3 Justificación metodológica	74
1.5.4 Justificación tecnológica	74
1.6 Hipótesis	74
1.6.1 Hipótesis general	74
1.6.2 Hipótesis específicas	75
1.7 Objetivos	76
1.7.1 Objetivo general	76
1.7.2 Objetivos específicos	76
II Marco metodológico	78
2.1 Variables	78
2.2 Operacionalización de variables	81
2.3 Metodología	83

2.4 Tipo de estudio	83
2.5 Diseño	84
2.6 Población, muestra y muestreo	84
2.6.1 Población	84
2.6.2 Muestra	84
2.6.3 Muestreo	85
2.7 Técnicas e instrumentos de recolección de datos	85
2.7.1 Técnica	85
2.7.2 Instrumentos	85
2.8 Métodos de análisis de datos	88
2.9 Aspectos éticos	88
III Resultados	90
3.1 Análisis de resultados	90
3.2 Resultados variable calidad	94
3.2.1 Análisis descriptivo variable Calidad Pretest Categorizado	94
3.2.2 Estadísticos variable Calidad Pretest	97
3.2.3 Análisis descriptivo variable Calidad Post test Categorizado	98
3.2.4 Estadísticos variable Calidad Post test	100
3.2.5 Prueba de Hipótesis variable Calidad Post test	102
3.2.6 Análisis de normalidad variable Calidad	104
3.3 Resultados variable performance	105
3.3.1 Análisis descriptivo variable Performance Pretest Categorizado	105
3.3.2 Estadísticos variable Performance Pretest	108
3.3.3 Análisis descriptivo variable Performance Post test Categorizado	109
3.3.4 Estadísticos variable Performance Post test	111
3.3.5 Prueba de Hipótesis variable Performance Post test	112
3.4 Resultados Variable Buenas practicas	116
3.4.1 Análisis descriptivo variable Buenas Prácticas Pretest categorizado	116
3.4.2 Estadísticos variable Buenas Prácticas Pretest	119
3.4.3 Análisis descriptivo variable Buenas Prácticas post test categorizado	120
3.4.4 Estadísticos variable Buenas Prácticas pos test	123
3.4.5 Prueba hipótesis variable Buenas Prácticas post test	124
3.4.6 Análisis de Normalidad	126

3.5	Resultados variable Malas Prácticas	128
3.5.1	Análisis descriptivo variable Malas Prácticas Pretest categorizado	128
3.5.2	Estadísticos variable Malas Prácticas Pretest	131
3.5.3	Análisis descriptivo variable Malas Prácticas post test categorizado	132
3.5.4	Estadísticos variable Malas Prácticas pos test	134
3.5.5	Prueba hipótesis variable Malas Prácticas post test	135
3.5.6	Análisis de Normalidad	137
3.6	Resultados variable Seguridad	139
3.6.1	Análisis descriptivo variable Seguridad pre test categorizado	139
3.6.2	Estadísticos variable Seguridad pre test agrupado	141
3.6.3	Estadísticos variable Seguridad pre test	141
3.6.4	Análisis descriptivo variable Seguridad pos test categorizado	142
3.6.5	Estadísticos variable Seguridad pos test	144
3.6.6	Prueba hipótesis variable Seguridad post test	145
3.6.7	Análisis de Normalidad	146
3.7	Resultados variable complejidad	147
3.7.1	Análisis descriptivo variable Complejidad pre test categorizado	147
3.7.2	Estadísticos variable Complejidad Pre test	150
3.7.3	Análisis descriptivo variable Complejidad post test categorizado	151
3.7.4	Estadísticos variable Complejidad pos test	154
3.7.5	Prueba hipótesis variable Complejidad post test	155
3.7.6	Análisis de Normalidad	157
3.8	Resultados variable Obsolescencia	159
3.8.1	Análisis descriptivo variable Obsolescencia Pre test Categorizado	159
3.8.2	Estadísticos variable Obsolescencia pre test	161
3.8.3	Análisis descriptivo variable Obsolescencia post test categorizado	162
3.8.4	Estadísticos variable Obsolescencia pos test	163
3.8.5	Prueba hipótesis variable Complejidad post test	164
3.8.6	Análisis de Normalidad	167
3.9	Resultados variable Mantenibilidad	169
3.9.1	Análisis descriptivo variable Mantenibilidad Pretest categorizado	169
3.9.2	Estadísticos variable Mantenibilidad Pretest	172
3.9.3	Análisis descriptivo variable Mantenibilidad post test Categorizado	173

3.9.4 Estadísticos variable Mantenibilidad post test	176
3.9.5 Prueba de hipótesis variable Mantenibilidad post test	177
3.9.6 Análisis de normalidad	178
3.10 Resultados generales	181
3.11 Resultados generales por variable	204
3.13 Relación de variable Calidad con los indicadores	236
IV Discusión	251
V.Conclusiones	253
VI. Recomendaciones	256
Primero:	257
VII. Referencias	258
VIII. Anexos	266

Indice de Tablas

Tabla 1 Descripción de nodos en el modelo de calidad RPG	24
Tabla 2. Evolution of Software Quality Models in Context of the Standard ISO 25010	30
Tabla 3.Tabla de Características de calidad en uso	41
Tabla 4. Tabla Características de calidad en producto	41
Tabla 5. Grupo de métricas	61
Tabla 6.Descripción de ítems de datos.	64
Tabla 7. Relación de Validadores	87
Tabla 8.Programas a analizar	91
Tabla 9.Datos Pre Test	92
Tabla 10.Datos Post Test	93
Tabla 11. Estadísticos variable calidad pre test.	94
Tabla 12. Estadísticos variable calidad pre test.	95
Tabla 13. Estadísticos variable calidad pre test.	96
Tabla 14.Tabla estadísticos calidad pre test.	96
Tabla 15. Estadísticos variable calidad pre test.	97
Tabla 16. Tabla estadísticos calidad pos test categorizado.	98
Tabla 17.Estadísticos calidad pos test categorizado.	98
Tabla 18.Tabla estadísticos calidad pos test.	100
Tabla 19. Tabla frecuencia calidad pos test.	101
Tabla 20.Estadísticas muestras emparejadas calidad pos test.	102
Tabla 21.Correlaciones muestras emparejadas calidad.	103
Tabla 22.Prueba T calidad pre test y pos test.	103
Tabla 23.Prueba de normalidad calida.	104
Tabla 24.Estadísticos variables performance_pretest (categorizado)	105
Tabla 25. Performance Pretest (categorizado)	106
Tabla 26.Estadísticos performance Pretest	106
Tabla 27. Performance Pretest	107
Tabla 28. Estadísticos performance Pretest	108
Tabla 29. Estadísticos performance pos test	109
Tabla 30.Performance pos test (categorizado)	109

Tabla 31. Performance Pos Test	110
Tabla 32. Estadísticos performance pos test	111
Tabla 33. Estadísticas muestras emparejadas PERFORMANCE	112
Tabla 34. Correlaciones muestras emparejadas PERFORMANCE	113
Tabla 35. Prueba de Muestras emparejadas PERFORMANCE	113
Tabla 36. Normalidad variable PERFORMANCE	114
Tabla 37. Pruebas de Normalidad PERFORMANCE	114
Tabla 38. Estadísticos B. Practicas Pre Test (agrupado)	116
Tabla 39. B. Practicas Pre Test (agrupado)	116
Tabla 40. Frecuencias B. Practicas Pre Test	118
Tabla 41. Estadísticos B. Practicas Pre Test	119
Tabla 42. Estadísticos B. Practicas Post Test (agrupado)	120
Tabla 43. B. Practicas Pos Test (agrupado)	120
Tabla 44. Frecuencias B. Practicas Pos Test	122
Tabla 45. Estadísticos B. Practicas Pos Test	123
Tabla 46. Estadísticas de Muestras emparejadas B. Practicas	124
Tabla 47. Correlación de muestras emparejadas B. Practicas	125
Tabla 48. Prueba de muestras emparejadas B. Practicas	125
Tabla 49. Resumen de procesamiento de casos B. Practicas	126
Tabla 50. Pruebas de normalidad B. Practicas	126
Tabla 51. Estadísticos Malas Prácticas (agrupado)	128
Tabla 52. Malas prácticas pre test (agrupado)	128
Tabla 53. Frecuencias Malas Prácticas Pre Test	130
Tabla 54. Estadísticos Malas Prácticas Pre Test	131
Tabla 55. Estadísticos Malas Prácticas Pos Test	132
Tabla 56. Malas prácticas Pos Test (agrupado)	132
Tabla 57. Frecuencias Malas Prácticas Pos Test	133
Tabla 58. Estadísticos Malas Prácticas Pos Test	134
Tabla 59. Estadísticos de muestras emparejadas Malas Practicas	135
Tabla 60. Correlaciones de muestras emparejadas Malas Practicas	136
Tabla 61. Prueba de muestras emparejadas Malas Practicas	136
Tabla 62. Resumen de procesamiento de casos Malas Practicas	137
Tabla 63. Pruebas de Normalidad Malas Prácticas	138

Tabla 64. Estadísticos seguridad pre test agrupado	139
Tabla 65 Frecuencias Seguridad pre test agrupado	140
Tabla 66 Estadísticos Seguridad pre test agrupado	141
Tabla 67 Frecuencia seguridad pre test agrupado	141
Tabla 68. Estadísticos seguridad pre test	141
Tabla 69 Estadísticos seguridad pos test agrupado	142
Tabla 70. Frecuencia seguridad pos test agrupado	142
Tabla 71. Frecuencia seguridad pos test	143
Tabla 72 Estadísticos seguridad pos test	144
Tabla 73 Estadísticas muestras emparejadas seguridad	145
Tabla 74 Resumen casos seguridad	146
Tabla 75. Estadísticos Complejidad Pre Test	147
Tabla 76. Frecuencia Complejidad Pre Test (agrupado)	147
Tabla 77. Frecuencias Complejidad Pre Test	149
Tabla 78. Estadísticos Complejidad Pre Test	150
Tabla 79. Estadísticos Complejidad Pos Test	151
Tabla 80. Frecuencia Complejidad Pos Test (agrupado)	151
Tabla 81. Frecuencias Complejidad Pos Test	153
Tabla 82. Estadísticos Complejidad Pos Test	154
Tabla 83. Estadísticas de muestras emparejadas Complejidad	155
Tabla 84. Correlaciones muestras emparejadas Complejidad	156
Tabla 85. Prueba de muestras emparejadas Complejidad	156
Tabla 86. Resumen de procesamiento de casos Complejidad	157
Tabla 87. Pruebas de Normalidad Complejidad	157
Tabla 88. Estadísticos Obsolescencia pre test	159
Tabla 89. Obsolescencia pre test agrupado	159
Tabla 90. Obsolescencia pre test	160
Tabla 91. Estadísticos Obsolescencia pre test	161
Tabla 92. Estadísticos pos test agrupado	162
Tabla 93. Obsolescencia pos test agrupado	162
Tabla 94. Estadísticos Obsolescencia pos test	163
Tabla 95. Estadísticas muestras emparejadas	164
Tabla 96. Correlación muestras emparejadas Obsolescencia	165

Tabla 97. Prueba muestras emparejadas Obsolescencia	166
Tabla 98	167
Tabla 99 Pruebas normalidad Obsolescencia	167
Tabla 100. Calidad pre test (agrupado)	169
Tabla 101. Mantenibilidad Pre Test (agrupado)	169
Tabla 102. Mantenibilidad Pre Test	171
Tabla 103. Estadísticos Mantenibilidad Pre Test	172
Tabla 104. Mantenibilidad Pos Test (agrupado)	173
Tabla 105. Mantenibilidad Pos Test (agrupado)	173
Tabla 106. Mantenibilidad Pos Test	175
Tabla 107. Estadísticos Mantenibilidad Pos Test	176
Tabla 108. Estadísticas de muestras emparejadas Mantenibilidad	177
Tabla 109. Correlación de muestras emparejadas Mantenibilidad	178
Tabla 110. Prueba de muestras emparejadas MANTENIBILIDAD	178
Tabla 111. Resumen de procesamiento de casos MANTENIBILIDAD	179
Tabla 112. Pruebas de Normalidad MANTENIBILIDAD	179
Tabla 113. Correlacion calidad e indicadores.	250

Indice de Figuras

<i>Figura 1.</i> ADG Modelo de calidad RPG. Obtenido de Ladányi, et al (2015).	26
<i>Figura 2.</i> Comparación de funciones de densidad de probabilidad. Obtenido de Ladányi, et al (2015).	28
<i>Figura 3.</i> Perspectiva histórica de los modelos de calidad de software. José P. Miguel (2014)	31
<i>Figura 4.</i> Modelo ISO / IEC 25010.	36
<i>Figura 5.</i> Marco de métricas de calidad de software. Obtenido de IEEE 1061-1998.	57
<i>Figura 6.</i> Calidad pre test categorizado.	95
<i>Figura 7.</i> Calidad pos test categorizado.	99
<i>Figura 8.</i> Calidad pre test y pos test.	103
<i>Figura 9.</i> Gráficos Q-Q normales calidad.	105
<i>Figura 10.</i> Perfomance pre test categorizado.	106
<i>Figura 11.</i> Perfomance pos test categorizado.	110
<i>Figura 12.</i> Perfomance pre y pos test.	112
<i>Figura 13.</i> Análisis normalidad perfomance.	115
<i>Figura 14.</i> Buenas prácticas pre test agrupado.	117
<i>Figura 15.</i> Buenas prácticas pos test agrupado.	121
<i>Figura 16.</i> Buenas prácticas pre y pos test.	124
<i>Figura 17.</i> Normalidad Buenas Prácticas.	127
<i>Figura 18.</i> Malas prácticas pre test agrupado.	129
<i>Figura 19.</i> Malas prácticas pos test agrupado.	133
<i>Figura 20.</i> Malas prácticas pre y pos test.	136
<i>Figura 21.</i> Normalidad Malas prácticas.	139
<i>Figura 22.</i> Seguridad pre test agrupado.	140
<i>Figura 23.</i> Seguridad pos test agrupado.	143
<i>Figura 24.</i> Normalidad seguridad.	146
<i>Figura 25.</i> Complejidad pre test.	148
<i>Figura 26.</i> Complejidad pos test.	152
<i>Figura 27.</i> Complejidad pre y pos test.	155

<i>Figura 28.</i> Normalidad Complejidad.	158
<i>Figura 29.</i> Obsolescencia pre test agrupado.	160
<i>Figura 30.</i> Obsolescencia pos test agrupado.	163
<i>Figura 31.</i> Obsolescencia pre y post.	165
<i>Figura 32.</i> Normalidad Obsolescencia.	168
<i>Figura 33.</i> Mantenibilidad pre test agrupado.	170
<i>Figura 34.</i> Mantenibilidad pos test agrupado.	174
<i>Figura 35.</i> Calidad pre y pos test.	177
<i>Figura 36.</i> Mantenibilidad de Calidad.	181
<i>Figura 37.</i> Mantenibilidad de Calidad.	182
<i>Figura 38.</i> Datos Pre Test de Calificaciones de las dimensiones.	184
<i>Figura 39.</i> Datos Pos Test de Calificaciones de las dimensiones.	185
<i>Figura 40.</i> Box plot de calificaciones pre – pos test para las dimensiones.	186
<i>Figura 41.</i> Histogramas y densidad pre y pos test.	187
<i>Figura 42.</i> Scatter Plots de calidad pre-post test.	189
<i>Figura 43.</i> Histogramas de calidad vs indicadores pre-post test.	191
<i>Figura 44.</i> Box plot general pre-post test.	193
<i>Figura 45.</i> Density plots pre-post test.	195
<i>Figura 46.</i> Histogramas de calidad e indicadores pre-post test.	197
<i>Figura 47.</i> Boxplots calidad vs indicadores post test.	199
<i>Figura 48.</i> Graficas polares de resultados generales.	201
<i>Figura 49.</i> Resultados Generales, gráficos de torta.	203
<i>Figura 50.</i> Análisis de calidad general.	204
<i>Figura 51.</i> Análisis de calidad general.	205
<i>Figura 53.</i> Análisis de calidad general.	206
<i>Figura 55.</i> Análisis de perfomance general.	208
<i>Figura 56.</i> Análisis de perfomance general.	209
<i>Figura 58.</i> Grafico de puntos pre y pos tes para variable perfomance.	210
<i>Figura 60.</i> Análisis de Buenas prácticas general.	212
<i>Figura 61.</i> Análisis de Buenas prácticas general.	213
<i>Figura 63.</i> Grafico de puntos pre y pos tes para variable buenas prácticas.	214
<i>Figura 65.</i> Análisis de malas prácticas general.	216
<i>Figura 66.</i> Análisis de malas prácticas general.	217

<i>Figura 68.</i> Grafico de puntos pre y pos tes para variable malas prácticas.	218
<i>Figura 70.</i> Análisis de complejidad general.	219
<i>Figura 71.</i> Análisis de complejidad general.	220
<i>Figura 73.</i> Grafico de puntos pre y pos tes para variable complejidad.	221
<i>Figura 75.</i> Análisis de seguridad General.	223
<i>Figura 76.</i> Análisis de seguridad General.	224
<i>Figura 77.</i> Grafico de frecuencias pre y pos tes para variable seguridad.	225
<i>Figura 78.</i> Grafico de puntos pre y pos tes para variable seguridad.	226
<i>Figura 79.</i> Análisis de Obsolescencia General.	228
<i>Figura 80.</i> Análisis de Obsolescencia General.	229
<i>Figura 82.</i> Grafico de puntos pre y pos tes para variable obsolescencia.	230
<i>Figura 84.</i> Grafico de análisis de mantenibilidad general.	232
<i>Figura 85.</i> Grafico boxplots de mantenibilidad general.	233
<i>Figura 87.</i> Grafico de líneas de pre tes y post test de mantenibilidad general.	234
<i>Figura 120.</i> Scatterplot de calidad y sus dimensiones.	237
<i>Figura 121.</i> Matriz de correlación de calidad y sus indicadores.	239
<i>Figura 122.</i> Matriz de correlación de calidad y sus indicadores.	240
<i>Figura 123.</i> Matriz de correlación de calidad y sus indicadores.	241
<i>Figura 124.</i> Matriz de correlación entre calidad y las dimensiones.	243
<i>Figura 125.</i> Histogramas pos test.	245
<i>Figura 126.</i> Matriz de correlación entre calidad y las dimensiones.	247
<i>Figura 127.</i> Matriz de correlación entre calidad y las dimensiones.	248
<i>Figura 128.</i> Matriz de correlación entre calidad y las dimensiones.	249
<i>Figura 149.</i> Modelo de calidad. Fuente Ladányi, et al (2015).	252
<i>Figura 150.</i> Modelo de calidad usado con QSOURCE.	252
<i>Figura 151.</i> Analizar programa interactivamente.	285
<i>Figura 152.</i> Estadísticas programa.	285
<i>Figura 153.</i> Estadísticas programa en fechas.	286
<i>Figura 154.</i> Consulta de estadísticas.	286
<i>Figura 155.</i> Resumen de estadísticas de programa 1.	287
<i>Figura 156.</i> Resumen de estadísticas de programa 2.	287
<i>Figura 157.</i> Resumen de estadísticas de programa 4.	288
<i>Figura 158.</i> Resumen de estadísticas de programa 5.	288

<i>Figura 159.</i> Resumen de estadísticas de programa 6.	289
<i>Figura 160.</i> Resumen de estadísticas de programa 7.	289
<i>Figura 161.</i> Resumen de estadísticas de programa 8.	290
<i>Figura 162.</i> Resumen de estadísticas de programa 9.	290
<i>Figura 163.</i> Resumen de estadísticas de programa 10.	291
<i>Figura 164.</i> Reportes de estadísticas.	291
<i>Figura 165.</i> Estadísticas generales.	292
<i>Figura 166.</i> Consulta general de calificación.	292

Resumen

Esta Investigación pre-experimental de una muestra tuvo como propósito fundamental determinar la influencia del uso del software QSOURCE en la mejora de la calidad del software desarrollado en lenguaje RPG en el departamento de sistemas del banco AGROBANCO. A una muestra de 31 programas se les analizó con la herramienta QSOURCE y su algoritmo de análisis estático de código generando calificaciones para cada uno de los indicadores de las 7 dimensiones de la variable.

La información para el análisis y la recolección de datos se realizó a través de un pre-test, y un pos-test usando la herramienta QSOURCE. El análisis estadístico se realizó aplicando el SPSS a un grupo experimental antes y después del tratamiento. También se le aplicó la prueba T de Student al grupo antes y después de la estrategia para comprobar la equivalencia inicial de ellos. La base teórica de la variable calidad del software se sustentó en las técnicas de revisión y aseguramiento de calidad de software de Pressman (2010), en el modelo de calidad del software ISO/IEC 2510 y en IEEE 1061-1998 - IEEE Standard for a Software Quality Metrics. La base teórica para la variable sistema QSOURCE se sustenta en las técnicas de revisión y aseguramiento de calidad de software de Pressman (2010), en el modelo de calidad de software para RPG de Ladányi, et al (2015), y en el soporte de métricas de software automatizado de Senevirathne, et al (2012). Los resultados obtenidos evidenciaron una mejora en la calificación promedio de la calidad de 54.48 a 66.16 en una primera iteración.

Se compararon los resultados para el grupo con pre y el pos-test permitiendo demostrar la hipótesis del trabajo y la efectividad del software para mejorar la calidad, a través del uso del software los programas desarrollados en RPG incrementaron sus calificaciones para los índices de performance, mantenibilidad, buenas prácticas, malas prácticas, complejidad y obsolescencia, como se evidencia en los resultados.

Palabras claves: Análisis estático de código, RPG, Calidad de software, CASE, Software.

Abstract

This pre-experimental investigation of a sample had as its fundamental purpose to determine the influence of the use of the QSOURCE software in the improvement of the software quality developed in RPG language in the systems department of the AGROBANCO bank. A sample of 31 programs was analyzed with the QSOURCE tool and its static code analysis algorithm, generating ratings for each of the 7 indexes of the variable.

The information for the analysis and data collection was made through a pre-test, and a post-test using the QSOURCE tool. The statistical analysis was performed applying the SPSS to an experimental group before and after the treatment. The Student's T test was also applied to the group before and after the strategy to check the initial equivalence of them. The theoretical basis of the software quality variable is based on Pressman's software quality review and assurance techniques (2010), the ISO / IEC 2510 software quality model and IEEE 1061-1998 - IEEE Standard for a Quality Metrics Software. The theoretical basis for the QSOURCE system variable is based on Pressman's software quality review and assurance techniques (2010), on the RPG software quality model of Ladányi, et al (2015), and on the metrics support of automated software from Senevirathne, et al (2012). The results obtained showed an improvement in the average quality rating from 54.48 to 66.16 in a first iteration.

The results were compared for the group with pre- and post-test allowing to demonstrate the work hypothesis and the effectiveness of the software to improve the quality, through the use of the software the programs developed in RPG increased their ratings for the performance indexes, maintainability, good practices, bad practices, complexity and obsolescence, as evidenced by the results.

Keywords: Static code analysis, RPG, Software quality, CASE, Software.

I Introducción

1.1 Realidad problemática

En ambientes bancarios en todo el mundo también en el Perú (banco Agrobanco, banco Banbif, banco Financiero entre otros), donde se usa la plataforma IBM/AS400 se tiene un problema de calidad en las aplicaciones legadas que están desarrolladas en lenguaje RPG que vienen desde la década de 1959 y tienen versiones antiguas del lenguaje donde el software desarrollado en los años subsecuentes sigue los lineamientos antiguos y obsoletos, este problema se ha venido dando hace mucho tiempo atrás por el desconocimiento de técnicas de Ingeniería de Software para mejorar las millones de líneas de código antiguo de las aplicaciones y por no existir herramientas de análisis estático que analicen características únicas del lenguaje RPG relacionadas con calidad.

Los millones de líneas de código de programas hechos en RPG actualmente en los bancos que usan IBM/RPG, tienen problemas de performance, al mantener códigos que hacen uso ineficiente de los recursos de CPU y memoria, igualmente existen código que impide la mantenibilidad de esos programas, existen códigos en desuso apuntando a malas prácticas, tienen pocas buenas prácticas de codificación implementadas, usan código y técnicas que hacen muy complejo los sistemas y poseen código con potenciales problemas de seguridad.

Todos estos detalles me sirvieron para desarrollar esta investigación que es necesaria para fortalecer las capacidades de toma de decisión de los gerentes de sistemas y de desarrollo de software sobre todo porque no tienen forma de acceder y visibilizar la información de calidad de sus sistemas de tal forma que les permita tomar medidas correctivas y de gestión. Es necesario poner a disposición de estas entidades de esta herramienta QSOURCE para mejorar la operativa diaria.

1.2 Trabajos previos

1.2.1 Trabajos previos internacionales

Artigas (2017) *Continuous integration and monitorization system for code quality and cooperation work*. Master's degree in Applied Telecommunications and Engineering Management. Universidad Politécnica de Cataluña. Cataluña. España.

El autor baso su investigación en el problema general de la calidad del código fuente de programas. El objetivo de esta tesis fue describir la configuración de un sistema de integración continua y un sistema de monitoreo con un software. El objeto de estudio, fueron los programas seleccionados para usarse en grupo. La metodología usada fue una investigación comparativa. El tipo de estudio es descriptivo. Diseño es experimental. La Población fue los programas de una empresa. El objeto fue 4 programas usados. Los instrumentos usados fueron las 4 herramientas de software Git, Bitbucket, Jenkins, SonarQube, Artifactory, Nagios. Como resultado se verifico que la cooperación entre equipos de desarrollo de software y el uso de herramientas de calidad de código mejora la calidad del proyecto.

Dankovčíková (2017) *Custom Roslyn Tool for Static Code Analysis. Master's degree in Informatics*. Masaryk University. Brno. República Checa. El autor partió del problema de la falta de herramientas de análisis de código estático para .NET. El objetivo de esta tesis fue crear una herramienta personalizada para análisis de código estático que verificaría el cumplimiento de las reglas internas de codificación y las pautas seguidas por los desarrolladores en Kentico Software, proporcionando una corrección de código automatizado cuando sea posible. El objeto fue la herramienta creada. La metodología usada fue una investigación descriptiva de la herramienta. El tipo de estudio es descriptivo. Diseño es experimental. La población fue los programas existentes de análisis de código. El instrumento utilizado fue el programa desarrollado que genero datos de calidad. Como resultado se obtuvo las mediciones de la herramienta sobre programas en la empresa real.

Ahuja (2017) *Robust Malware Detection Using Integrated Static and Dynamic Analysis*. Degree of Master Technology. Indian Institute of Technology Kanpur. Kanpur India. El autor se centró en el problema de la necesidad de herramientas para detección de malware. El objetivo fue el de integrar las técnicas de análisis estático y dinámico, usar técnicas de machine Learning para detección eficiente de malware y construir modelos predictivos. El objeto de la investigación fue los métodos de análisis de código y análisis de datos. La metodología que se uso fue la investigación bibliográfica de fuentes sobre el tema y la creación de un modelo o

método de detección de malware, la recolección de 15.000 malware, y el posterior análisis. El tipo de estudio es descriptivo. Diseño es experimental. La población fue los métodos de análisis de código, la técnica machine Learning. El instrumento fue la herramienta Virus Total Intelligence, que generó datos de 15.000 malware. Los resultados obtenidos muestran que el método híbrido, donde se usan los métodos estático y dinámico de análisis de código, genera buenos resultados con el enfoque gradientboosting.

Bán (2017) *Static source code analysis in pattern recognition, performance optimization and software maintainability*. Degree of doctor of philosophy. University of Szeged. Hungary. El problema en esta tesis, se centró en dos temas principales, los efectos de los patrones de código fuente en la mantenibilidad del software y la utilización de métricas de código fuente estáticas en la optimización del rendimiento. El objetivo fue llamar la atención sobre la importancia de la fase de mantenimiento e ilustrar sus activos y riesgos al resaltar los patrones de diseño y sus efectos tangibles y los anti patrones pueden tener en la mantenibilidad del software y ayudar a los desarrolladores a utilizar más fácilmente el hardware acelerador moderno y aumentar el rendimiento creando una metodología extensible y fácilmente utilizable para construir modelos de selección de plataforma estática. La metodología fue analizar la relación entre los patrones de diseño y la capacidad de mantenimiento, se calcularon medidas para cada revisión en JHotDraw.

El tipo de estudio es descriptivo. Diseño es experimental. La población fue las herramientas de análisis de software. El instrumento fueron las herramientas de análisis estático y estadísticas. Los resultados encontraron una relación proporcional entre los patrones de diseño y mantenibilidad, una correlación positiva entre anti patrones y fallas del programa. En el campo de la optimización del rendimiento, se demostró la metodología de creación de modelos de predicción de plataformas tanto cualitativas como cuantitativas, que dependen de la información estática. El resultado principal es la metodología en sí misma, junto con sus detalles y un estudio complementario que demuestra el efecto perjudicial de la paralelización de código fuente en mantenibilidad.

Ramos (2016) *Diseño de un modelo de evaluación de la calidad de productos de software, basado en métricas externas y usabilidad aplicado a un caso de estudio*. Máster en gestión de tecnologías de información. Escuela Politécnica Nacional. Quito. Ecuador. El autor se centró en el análisis de la situación actual de la problemática de la calidad de software y la determinación de características funcionales de los productos de software. Conocer el nivel de calidad del producto de software en el caso de estudio a evaluar. Permitir la medición de la calidad del sistema mediante el modelo de evaluación de calidad de productos de software. La metodología fue una investigación comparativa de modelos de calidad existentes. El tipo de estudio es descriptivo. Diseño es experimental. La población fue los programas de video juegos. El objeto fue un caso de estudio de un software de video juegos. El instrumento es un formulario con parámetros para medir la calidad. Como resultado se probó el modelo propuesto en el caso de estudio y se obtuvieron los cálculos de las métricas cuyos resultados son parte de las recomendaciones.

Shrestha (2016) *Best QMEs for measurement of Software quality for SMEs*. Master's Thesis Degree Programme in Computer Science and Engineering. University of Eastern Finland, Faculty of Science and Forestry, Joensuu School of Computing. Joensuu, Finland. El autor baso su investigación en la falta de elementos de medición de la calidad del software. El objetivo fue identificar los mejores elementos para medición de calidad de software, Quality Measure Elements (QME) usando estándares ISO/IEC. El objeto de investigación fueron las normas ISO/IEC para SMEs. La metodología que se uso fue la realización de encuestas a empresa. El tipo de estudio es descriptivo. Diseño es no experimental. La población fueron las empresas con desarrollo de software, y el objeto fueron las 3 empresas seleccionadas. (Measurement of Software Quality for SMEs). El instrumento utilizado fue un cuestionario de preguntas. Como resultado se obtuvo una lista de elementos de medida de calidad más usados en las empresas del estudio.

Taipale (2015) *Improving software quality with software error prediction*.

Master's Thesis Degree Programme in Computer Science and Engineering. University of Oulu. Oulu. Finland. El autor baso su investigación en el problema de la predicción de errores. El objetivo describir herramientas y prácticas de desarrollo de software para la creación de un sistema para predecir errores. La metodología usada fue la investigación de fuentes bibliográficas. El tipo de estudio es descriptivo. Diseño es experimental. La población fue las herramientas y teorías actuales sobre desarrollo, prueba y predicción de errores en software. El instrumento fue la herramienta desarrollada. Como resultado se describió los pasos para crear un predictor de errores, se creó la herramienta y se pudo obtener buenas predicciones de los errores.

Rakic (2015) *SSQSA Set of Software Quality Static Analysers*. University of NoviSad Faculty of sciences department of mathematics and Informatics. NoviSad. Serbia. El autor baso su investigación en el problema de la aplicación de métricas de software y otras técnicas de análisis estáticos que tienen limitaciones e inadecuadas herramientas involucradas. El objetivo fue desarrollar el marco SSQSA para el monitoreo y control consistente de la calidad del software a nivel de código. La metodología fue la descripción comparativa de cómo se aplica una métrica en diferentes lenguajes. Esta exploración se repitió en un amplio espectro de métricas y la exploración comparativa en la aplicación de un amplio espectro de métricas en un solo lenguaje, esta observación fue repetida en todos los lenguajes incluidos en la investigación. El tipo de estudio es descriptivo. Diseño es experimental. La población fue las métricas de software y los lenguajes de programación existentes. La muestra de lenguajes fue: Java, C#, Delphi, COBOL, C, Pascal, Modula-2, Scheme, Erlang, Python, PHP, JavaScript, OWL, WSL, and Tempura. El instrumento utilizado fue, Set of Software Quality Static Analyzers (SSQSA). Se logró describir el grupo de analizadores estáticos de calidad de software (SSQSA), se pudo generar la representación intermedia del código analizado y se usó analizadores estáticos específicos de cada lenguaje analizado se tabularon los resultados de análisis de calidad.

Hietala (2014) *Assisting Software Quality Assurance by Change Impact Analysis*. Master's Degree programme in InformationTechnology. Jyväskylä,

Finlandia. El autor baso su investigación en el cambio en los programas y la necesidad de una herramienta de análisis de impacto de cambios. El objetivo fue el desarrollo de una herramienta para verificar los cambios usando técnicas de análisis de código estático. El objeto de la investigación fue el software desarrollado. Como metodología se propuso el modelo de identificar el problema, motivación, objetivos de la solución, diseño, desarrollo y demostración. El tipo de estudio es descriptivo. Diseño es experimental. La población fue las herramientas existentes en el mercado. El instrumento fue la herramienta desarrollada que género los datos del análisis. Como resultado, se obtuvo una herramienta que permite visualizar los cambios del software usando técnicas de análisis de código estático para mejorar la calidad.

Hegedus (2014) *Advances in Software Product Quality Measurement and its Applications in Software Evolution*. University of Szeged. Hungary. El investigador baso su investigación en la creciente dependencia de los sistemas de software y su calidad y fiabilidad. El objetivo es proporcionar una medida de alto nivel para la mantenibilidad, elaborar métodos para adquirir información útil de bajo nivel sobre la mantenibilidad a nivel de los elementos del código fuente que pueden usarse para mejorar el Mantenimiento del Sistema. La metodología se basó en examinar dos sistemas Java con el nuevo modelo probabilístico de calidad y se hizo una encuesta a expertos y el nuevo enfoque probabilístico se implementó en una herramienta llamada SourceAudit, como parte de un marco de monitoreo de calidad continuo llamado QualityGate. El tipo de estudio es descriptivo. Diseño es experimental. El autor realizó una encuesta sobre modelos prácticos existentes y exploró sus antecedentes teóricos. Realizó una validación empírica de un nuevo método de evaluación de la calidad probabilística, evaluado los resultados e implementado los prototipos de herramientas que respaldan la validación empírica.

1.2.2 Trabajos previos nacionales

Espejo (2016) *Modelo de aseguramiento de la calidad en el proceso de desarrollo*

de software basado en los modelos de madurez de capacidades (CMMi), proceso de software para equipos (TSP) y personas (PSP). Magister en Gobierno de Tecnologías de Información. Universidad Nacional Mayor de San Marcos. Lima. Perú. El autor definió su problema como, ¿De qué manera la integración de un modelo basado en los modelos de madurez de capacidades integrado, procesos de software de equipos y proceso de software de personas permite asegurar la calidad en el proceso de desarrollo de software? el objetivo general fue generar un modelo que permita asegurar la calidad en el proceso de desarrollo de software basado en los modelos de madurez de capacidades, proceso de software de equipos y proceso de software de personas. El objeto de investigación fue los modelos de calidad de software. El tipo de investigación es cuantitativa experimental. El diseño es pre experimental con prueba antes y después a un solo grupo. La población fue los proyectos de desarrollo de software que ejecutan las consultoras y empresas (in house) en el departamento de Lima – Perú, cuyo proceso de desarrollo de software se encuentra certificado por CMMI Institute en el Nivel de Madurez 2 o 3. Se usó muestreo no probabilístico con una muestra de 22 proyectos de software. Como instrumento se recopiló datos a través de tesis, direcciones de internet, encuestas diseñadas para este propósito, documentación de proyectos (Líneas base, Trabajo real, etc.), resultados de evaluaciones a sus procesos y reportes de la oficina de gestión de proyectos (PMO) de las empresas/consultoras. Se concluye que el modelo de aseguramiento de calidad tuvo un impacto positivo en el proceso de desarrollo de software porque contribuye en el aseguramiento de la calidad en el proceso de desarrollo.

Baldeón (2015) *Método para la evaluación de calidad de software basado en ISO/IEC 25000.* Maestro en computación y sistemas con mención en gestión de tecnologías de información. Universidad San Martín de Porres. Lima Perú. El autor basó su investigación en la calidad insuficiente en los productos de software, planteando la pregunta, ¿De qué manera un método para la evaluación de calidad mejorará la calidad del software? el objetivo fue mejorar la calidad del software a través de la aplicación de un método para la evaluación de calidad basado en ISO/IEC 25000. El objeto de investigación fue el estándar ISO/IEC 25000. La metodología consistió en selección del diseño experimental a seguir, selección de

la muestra, recolección de los datos, análisis de los datos obtenidos. El tipo de estudio es descriptivo. Diseño es experimental. El diseño seleccionado es cuasi experimental con pos prueba. La población son los proyectos de la Financiera Autos de Perú, con duración de entre tres y nueve semanas cronológicas (desde el análisis hasta la conformidad del usuario) y cuya fecha de inicio es mayor al 01/01/2014. La muestra para este estudio son 28 proyectos divididos en dos grupos. El instrumento fue la revisión de la Bitácora de pases al ambiente de aceptación por el usuario, Revisión del Reporte de errores identificados. Se logró mejorar la calidad del producto software como resultado de la aplicación del método de evaluación de calidad de software basado en ISO/IEC 25000.

Huanca (2015) *Revisión sistemática de la calidad del software en prácticas ágiles*. Magíster en Informática con mención en Ingeniería del Software. Pontificia universidad católica del Perú. Lima. Perú. El autor definió su problema y se planteó en dos preguntas, ¿cuál es el estado actual del conocimiento acerca de la calidad en las prácticas ágiles?, ¿cuáles son las prácticas más importantes para el logro de calidad en el desarrollo ágil? Los objetivos fueron: comprender los conceptos de calidad y metodologías ágiles, resumir el estado del arte de las investigaciones existentes relacionadas a estudios empíricos relativos a la calidad en las prácticas ágiles, identificar las lagunas de investigación existentes en el área de calidad en metodologías ágiles, con el fin de sugerir temas para investigaciones futuras. El objeto de la investigación fueron los 61 estudios que pasaron el proceso de evaluación de la calidad. La metodología se basó en revisión sistemática de la literatura. El tipo de estudio es descriptivo. Diseño es no experimental. La población fue toda la literatura sobre calidad. Como resultado se obtuvo una mejora medible de la calidad, y resultados más favorables aun al utilizarlas en equipos pequeños. La productividad mostró un considerable incremento reflejándose en una mayor calidad del código generado y una menor cantidad de defectos.

1.3 Teorías relacionadas al tema

1.3.1 Sistema QSOURCE

Dado que QSOURCE es un sistema de software, se define un sistema basado en computadora como un conjunto o disposición de elementos que están organizados para realizar un objetivo predefinido procesando información. (Pressman, 2010, p.166). Se define el software como el producto que diseñan y construyen los ingenieros de software, esto abarca programas que se ejecutan dentro de una computadora de cualquier tamaño y arquitectura, documentos que comprenden formularios virtuales e impresos y datos que combinan números y texto y también incluyen representaciones de información de audio, video e imágenes. (Pressman, 2010, p3)

Ladányi, et al (2015) publicaron un paper de la Investigación "A software quality model for RPG". In 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER) (pp. 91-100). IEEE. Los investigadores crearon un modelo de calidad basado en el estándar ISO/IEC25010. La principal contribución de este estudio es la creación de un enfoque de gestión de calidad continuo especializado para RPG. Investigaron el estado del arte en las técnicas de evaluación de calidad para RPG, para explorar el código RPG y detectar componentes con alto riesgo, para ello usaron la herramienta SourceMeter para análisis de código estático. El analizador proveyó 3 tipos de medición: métricas de software, violación de reglas y duplicación de código.

Los investigadores usaron medidas de bajo nivel como entrada para proveer un diagnóstico sobre un sistema RPG. Este diagnóstico caracteriza la calidad de todo el sistema RPG está basado en el estándar ISO/IEC 25010. Como validación del enfoque muestran como en un caso de estudio gestionaron la integración del método integrándolo en los procesos de desarrollo y como fue usado para propósito de refactoring. Los investigadores comentan que una vez se tienen indicadores de calidad, como las métricas del software, que son capaces de caracterizar la calidad del software desde diferentes puntos de vista, es necesario estandarizar esta información. Usaron la herramienta y calcularon diferentes métricas para elementos de código fuente localizados en el código fuente, las métricas se organizaron en 4 grupos: (1) Size (LOC, LLOC) , (2) Documentation (CLOC, CD) , (3) Coupling (NII, NOI) , (4) Complexity (NLE, McCC) , (5). Según los investigadores el uso del modelo fue un éxito, integraron el modelo a una herramienta llamada QualityGate,

y en asociación con una empresa de desarrollo de software la probaron. Esta otra investigación da apoyo al uso de herramientas para el análisis de calidad en software desarrollado en RPG como QSOURCEy demuestra el uso de una herramienta como instrumento de medición y generación de datos de calidad usando un amplio rango de métricas, para la investigación.

Senevirathne, et al (2012) publicaron un paper: *Metric Based Code AnalyzingTool*. El objetivo principal de la publicación fue proveer un soporte de métricas de software automatizado para usuarios que analicen código fuente de software. Esta herramienta evalúa la calidad del software y formula estimaciones de acuerdo a un modelo basado en métricas jerarquizadas especificadas, que genera comentarios de calidad, análisis de resultados, revisión y resolución basados en resultados de métricas. Dice el autor que actualmente no hay un modelo estructurado para medir el código fuente que sea predictivo y confiable, sin embargo, existen muchas herramientas de análisis de código para asistir en la medición de software en la industria, la mayoría de ellas simplemente recolecta métricas no estructuradas, pero no proveen un enfoque bien definido para relacionar esas métricas a atributos externos de calidad del software.

Los investigadores han gestionado este problema vía la herramienta de análisis de código basado en métricas la cual es diseñada bajo cuatro categorías de métricos principales como orientados a objeto, orientado a complejidad, orientado a tamaño y métricas de mantenibilidad. Aunque estos investigadores usaron una herramienta para analizar código de un lenguaje orientado a objetos, este modelo sirve de base para herramientas de análisis de código de lenguajes procedurales como RPG, que es el objetivo de QSOURCE. Los investigadores buscaron las herramientas más usadas como: (1) Visual Studio Code MetricsPower Tool, (2) eclipse Metricsplug-in, (3) resource Standard Metrics, (4) justCode, (5)NDepend.

Ladányi, et al (2015) evaluaron estas herramientas e identificaron que tienen muchas debilidades como pobre consistencia y confiabilidad, complejidad de las mediciones, manera no estructurada de organizar las métricas. Se basaron en IEEE Standard 1061, para una metodología de métricas de calidad de software, el

cual presenta 3 marcos que soportan las métricas de software, y actividades de medición, (1) The Software Quality Metrics Methodology. (2) The Goal, Question, Metric (GQM) Paradigm, (3) Practical Software Measurement (PSM).

Las tecnologías y herramientas usadas fueron: (1) Visual Studio 2010, (2) Net Framework 4.0, (3) Microsoft SQL Server 2008, (4) Antlr, the Open Source Parser. Para implementar esta herramienta se siguió el enfoque basado en el estándar IEEE 1061, para seguir la conformidad de la industria de ingeniería de software. Probaron la herramienta con 25 códigos fuente de una variedad de aplicaciones, se usaron casos de prueba para evaluar la herramienta. Los resultados generados tuvieron 98% de precisión, construyeron una herramienta basada en cálculo de métricas para desarrolladores y gerentes de proyectos, los cuales podrán ejecutar sus funciones más efectivamente.

El autor comenta que el análisis estático examina el código de los programas y las razones de los posibles comportamientos que pueden ocurrir en tiempo de ejecución. Las herramientas basadas en análisis estático pueden ser usadas para detectar defectos en programas. En los comienzos del desarrollo de software no había conciencia de cuan necesario y efectivo puede ser una revisión del código, pero en los 1970, la revisión formal y las inspecciones de código se reconocieron como importantes para la productividad y la calidad del código y fueron adoptadas en proyectos de desarrollo de software. Este nuevo enfoque de desarrollo de software permite remover defectos en las etapas tempranas del proceso de desarrollo de software para producir programas más eficientes y confiables. El enfoque de análisis estático revisa el código fuente verificando la adherencia del código a ciertas reglas y parámetros. Gomes, et al, (2011).

QSOURCE es una herramienta de software que permite a los ingenieros de software, desarrolladores y gerentes de desarrollo, visibilizar y cuantificar la calidad de sus sistemas desarrollados en RPG, y así poder tomar decisiones que les permitan alinear sus procesos a los objetivos del negocio, en cuanto a la generación de valor. QSOURCE está desarrollado en lenguaje RPG, y funciona en la plataforma IBM/AS400 o I SERIES, permite definir métricas de calidad a validar en las fuentes de los programas evaluados, se definen las librerías donde se encuentran las

fuentes de programas RPG (RPG III, IV, FREE, SQLRPGLE, RPGLE) y se procede a analizar las fuentes de manera manual uno a uno o de manera masiva, al final se genera los resultados del análisis ya sea por pantalla o por reporte.

Esta herramienta analiza cada código fuente y verifica línea a línea si se encuentra presente uno de los tokens o palabras claves que están asociadas a uno de los parámetros de calidad parametrizados, se contabilizan y al final se calcula el total de tokens presentes en el código fuente, asociados a cada parámetro de calidad, los valores van de 1 a 100, al final el valor asociado determina la calificación de calidad de 1 a 40 bajo, de 41 a 70 medio y de 71 a 100 alto. Se entiende la palabra token como un componente léxico o símbolo, la primera fase de un compilador se llama análisis léxico, el analizador léxico lee la cadena de caracteres que hace parte del código fuente del Programa y agrupa los caracteres en secuencias llamadas lexemas, para cada lexema el analizador sintáctico genera como salida un token, en el token el primer componente es un símbolo abstracto que se usa durante el Análisis sintáctico. Aho, Lam, Sethi, Ullman (2007, pag. 6).

Los parámetros de calidad del código RPG se categorizan en performance, mantenibilidad, buenas prácticas, malas prácticas, seguridad, complejidad y obsolescencia. Un token puede estar asociado a varios parámetros. Cada token tiene un rango de 0 a n, que determina el número de apariciones validas esperadas. El enfoque de QSOURCE es adoptar este grupo de factores de calidad como una lista de comprobación para evaluar la calidad del software. A medida que se usa para medir la calidad se puede determinar si se está mejorando. El sistema propuesto se presenta en el Anexo 2.

QSOURCE es una herramienta CASE, sobre el término se tiene que: Las herramientas CASE (ComputerAided Software Engineering), ayudan a los gestores y practicantes de ingeniería del software en todas las actividades asociadas a los procesos de software. Automatizan las actividades de gestión de proyectos, gestionan todos los productos de los trabajos elaborados a través del proceso, y ayuda a los ingenieros en el trabajo de análisis, diseño y codificación. Las herramientas CASE se pueden integrar dentro de un entorno sofisticado. Pressman (2010, P.597).

Pressman (2010) hablo de métricas de navegación para abordar la complejidad del flujo de navegación en páginas web, y recomienda el uso de web StaticAnalyzersTool, para validar el código HTML contra lineamientos de usabilidad típicos. El autor habla del aseguramiento de la calidad del software como una tarea vital que debe tener metas pragmáticas como la calidad de los requerimientos, la calidad del diseño, la calidad del código.

BinKley (2007 p.1) , expuso muy bien la importancia de herramientas de ayuda para el análisis y el control de la calidad, el crecimiento exponencial de los sistemas de software y el número de líneas de código, el investigador refirió que en 25 años la base de código fuente global llegara a 500 billones de líneas de código (40%-60% son RPG y COBOL), debido a esto se debe usar herramientas que gestionen la complejidad y calidad y las hagan visibles cuantitativamente para dar soporte a la toma de decisiones en la operativa diaria de los ambientes de sistemas . Para entender la calidad del software debemos citar el concepto que nos brindan varios autores.

Importancia de la herramienta QSOURCE en Banco Agrario

El Banco Agrario en Perú tiene sedes descentralizadas en diferentes regiones y da servicios a más de cien mil usuarios, los servicios financieros permiten dar préstamos a los pequeños agricultores, el banco tiene su plataforma de sistemas de información en un servidor IBM/AS400, donde sus principales sistemas de información se encuentran desarrollados en lenguaje RPG, dado que sus más de 5.000 programas deben ser mantenidos según las necesidades, la calidad de los mismos debe de mantenerse adecuadamente para reflejarla en el servicio a los clientes internos y externos, aquí es donde QSOURCE da soporte, ya que permite visibilizar cuantitativamente la calidad de sus sistemas escritos en RPG, y así los jefes y gerentes de sistemas y desarrollo tienen información de primera mano para la toma de decisiones, que de otra manera sería imposible.

Calidad de software

Conceptos de autores referentes

Tian (2007) Preciso que la pregunta, "¿Qué es la calidad del software?", está destinada a generar muchas respuestas diferentes, dependiendo de a quién le pregunte, en qué circunstancias, para qué tipo de sistemas de software, y así. una pregunta alternativa que tal vez sea más fácil para obtener más informaciones "¿Cuáles son las características del software de alta calidad?, se define la calidad del software definiendo las características esperadas o propiedades del software de alta calidad. También se debe examinar las diferentes perspectivas y expectativas de los usuarios, así como de otras personas involucradas en el desarrollo, gestión, comercialización y mantenimiento de los productos de software. También se debe examinar las características individuales asociadas con la calidad y su interrelación, y se debe centrar en las características críticas de la corrección funcional.

Pressman (2010) definió la calidad de un sistema, aplicación o producto como el nivel o resultado de los requisitos que describen el problema, el diseño que modela la solución, el código que produce a un programa ejecutable, y las pruebas que permiten que el software detecte errores. Refirió que El *American Heritage Dictionary*, define la calidad como «una característica o atributo de algo». Como un atributo de un elemento, la calidad se refiere a las características medibles -cosas que se pueden comparar con estándares conocidos como longitud, color, propiedades eléctricas, maleabilidad, etc.-. Sin embargo, el software en su gran extensión, como entidad intelectual, es más difícil de caracterizar que los objetos físicos. No obstante, sí existen las medidas de características de un programa. Entre estas propiedades se incluyen complejidad ciclomática, cohesión, número de puntos de función, líneas de código y muchas otras.

Pressman (2010) define calidad de software como "Proceso eficaz de software que se aplica de manera que crea un producto útil que agrega valor medible a

quienes lo crean y a quienes lo usan". Pressman (2010) Planteo que las dimensiones y factores de la calidad (corrección, confiabilidad, eficiencia, integridad, usabilidad, flexibilidad, portabilidad, reusabilidad, mantenibilidad) se centran en el software como un todo y pueden utilizarse como indicación general de la calidad de una aplicación. El autor planteo que un equipo de software puede desarrollar un conjunto de características de la calidad y las preguntas asociadas correspondientes que demuestren el grado en el que se satisface cada factor.

IEEE Stándars Association (2016) definió la calidad del software como el grado en que el software posee una combinación deseada de atributos de calidad. Se definió el propósito de las métricas del software como realizar evaluaciones a lo largo del ciclo de vida del software sobre si los requisitos de calidad del software se están cumpliendo. El uso de métricas de software reduce la subjetividad en la evaluación y control de la calidad del software al proporcionar una base cuantitativa para tomar decisiones sobre la calidad del software. La metodología de métricas de calidad de software es un marco de IEEE Standard 1061 y fue diseñado para ser flexible. Este comienza con el establecimiento de requerimientos de calidad, las cuales son usadas para describir la calidad del software.

Según la (ISO / IEC 25010) calidad del software se entiende como el grado en que un producto de software satisface las necesidades declaradas e implícitas cuando se utiliza bajo especificación condiciones. Según la (ISO / IEC 25010) el concepto de performance se define como la cantidad de recursos usados bajo ciertas circunstancias, comportamiento del uso de recursos en el tiempo. El concepto de mantenibilidad lo define el según la (ISO / IEC 25010) como el grado de efectividad o eficiencia con la cual un producto o sistema puede ser modificado por mantenedores. El concepto de seguridad lo define el (ISO / IEC 25010) como el grado en que un producto o sistema protege la información y los datos para que las personas u otros productos o los sistemas tienen el grado de acceso a datos apropiado para sus tipos y niveles de autorización.

IBM (2012) especifica buenas prácticas como las instrucciones nuevas que deben ser usadas para reemplazar las antiguas y malas prácticas como las instrucciones antiguas de versiones anteriores del RPG. Según el glosario ISO/IEC/IEEE (2009), la complejidad se puede definir Como: 1. el grado en que el diseño o código de un sistema es difícil de entender debido a numerosos componentes o relaciones entre los componentes. 2. perteneciente a cualquier conjunto de métricas basadas en la estructura que miden el atributo en (1) 3. El grado en que un sistema o componente tiene un diseño o implementación que es difícil de entender y verificar.

Ladányi, et al (2015) explico que hay muchos sistemas legados escritos para ordenadores mainframe. Estos sistemas incluyen elementos críticos como procesamiento de datos a granel, estadísticas y manejo de transacciones. En 1988, IBM presentó la robusta plataforma AS / 400, que se hizo muy popular hasta finales de siglo (fue más tarde renombrado como IBM i). Tiene su propio entorno de programación llamado ILE (Integrated Language Environment), que permite utilizar programas escritos en idiomas compatibles con ILE, como el lenguaje de programación RPG (Reporting Program Generator).

Aplicaciones comerciales desarrolladas para la plataforma IBM usualmente usan el lenguaje de programación RPG de alto nivel. A pesar del hecho de que el lenguaje fue introducido en 1959, RPG es todavía ampliamente utilizado hoy en día, y está en constante evolución. Se encuentra cerca de los sistemas de bases de datos y generalmente se usa para procesar gran cantidad de datos.

La principal contribución en este trabajo es la introducción de un enfoque de gestión de calidad continua especializado en RPG. De acuerdo con nuestro mejor conocimiento, ningún o muy poco esfuerzo se invirtió en la investigación de la garantía de calidad con técnicas de última generación para RPG. Para explorar el código fuente de los RPG y detectar componentes que conllevan altos riesgos, utilizaron el SourceMeter como la herramienta del analizador de código estático RPG. El analizador proporciona tres tipos de mediciones, que son las siguientes:

Métricas de software: una métrica de software en este contexto es una medida de alguna propiedad del código fuente.

Ladányi, et al (2015) explicaron que existe una necesidad creciente en el desarrollo de software para definir medidas cuantitativas y objetivas, las métricas de software se calculan para elementos de lenguaje RPG (a saber, para subrutinas, procedimientos, programas, y para todo el sistema). Violaciones de reglas: violaciones de reglas pueden revelar segmentos de código que son propensos a errores. Estos pueden ser problemas de codificación que se introducen accidentalmente o debido a programadores de poca habilidad y código, pueden ser síntomas en el código fuente que posiblemente significa un problema de diseño más profundo en el código y puede causar un funcionamiento incorrecto en una etapa posterior. El código apestoso no es igual a los errores, pero la presencia de ellos está aumentando el riesgo de errores o fallas en el futuro. Las reglas se definen para RPG para indicar la existencia de posibles códigos apestosos.

Además de la codificación, las reglas son excelentes medios para definir estándares de codificación de la compañía. Duplicaciones de código: los segmentos de código duplicados vienen por lo general del muy productivo, pero al mismo tiempo peligroso hábito de copiar-pegar código fuente. Las duplicaciones de código podrían manejarse como un código apestoso; sin embargo, cubren una importante parte separable de aseguramiento de la calidad.

Utilizaron estas mediciones de bajo nivel como una entrada para proporcionar un diagnóstico acerca de un sistema RPG. Este diagnóstico, que caracteriza la calidad de todo un sistema, es provisto por la nueva calidad basada en el modelo estándar ISO / IEC 25010 presentado en este documento. Construyendo y probando una calidad, un modelo como este necesita especialistas en este dominio específico. Así que, la compañía llamada R & R Software1 (empresa mediana con más de 100 desarrolladores de software) se involucró para ayudar al calibración y evaluación del modelo de calidad con su profundidad conocimiento en el desarrollo de sistemas de software escritos en RPG.

Ladányi, et al (2015) aclararon la importancia de cada aspecto de calidad y también proporcionó una colección de programas industriales escritos en RPG. Como una validación de su enfoque, mostraron un caso de estudio de cómo lograron integrar su método en su desarrollo procesos, y cómo lo usaron para fines de refactorización. En las últimas décadas se han publicado varios estudios lidiando con métricas de software.

Como principio, Chidamber y Kemerer (1994 citado por Ladányi, et al (2015)) proporcionó varias definiciones de métricas orientadas a objetos. A pesar de que RPG no es un idioma de programación orientado a objetos, este estudio es esencial para futuras investigaciones en el área de métricas de software. Chidamber y Kemerer también desarrolló una herramienta de recopilación automatizada de datos para recopilar métricas de software. Después de recopilar métricas, sugirieron formas en que los gerentes pueden usar estas métricas para mejorar los procesos.

Se realizaron muchos esfuerzos de investigación para encontrar una estrategia para migración efectiva de sistemas heredados. Canfora et al. Presentaron una estrategia de migración incremental para transformar programas legados en un entorno orientado a objetos. La estrategia de migración consiste en seis fases secuenciales, tratando de identificar objetos en un ambiente persistente de almacenamiento de datos. Aplicando el proceso de migración se utiliza principalmente para transformar RPG II y Programas RPG III en RPG IV.

Las métricas de software generalmente se implementan para la mayoría lenguajes de programación ampliamente utilizados como C, C ++, Java y C #. Solo unos pocos estudios se ocupan de la garantía de calidad del lenguaje RPG. En 1982 Hartman publicó un estudio que trataba con la identificación de errores de software en RPG II y RPG III que usan las métricas de complejidad de Halstead y McCabe métrica. El estudio fue evaluado y esta comercialmente disponible y los módulos escritos en IBM RPG. Usando medidas, los módulos se asignaron a un valor de métrica bajo, medio o alto rangos. Concentrarse en los módulos que caen en el medio y altos rangos de valores métricos aumentó la efectividad de encontrar la mayoría de los defectos en todo el sistema. La complejidad de las métricas de

McCabe parecía un poco mejor en la identificación de módulos con errores que la métrica de complejidad de Halstead.

Otro papel temprano maneja la tasa de error de un software como una medida de calidad del código. Naib desarrolló una metodología para la predicción de la tasa de error y, por lo tanto, la calidad del código antes del lanzamiento de una aplicación. Naib consideró las mismas métricas (mencionadas como factores internos) como Hartman, a saber, complejidad Halstead (1976 citado por Ladányi, et al (2015)) y McCabe (1977 Ladányi, et al (2015)) y también líneas de código (LOC). Al contrario de ellos, el autor no usa métricas de software para identificar errores en el sistema, pero proporciona un enfoque flexible que es capaz de proporcionar varios índices de calidad.

Kan et al. (1994 citado por Ladányi, et al (2015)) estaban tratando con la gestión de la calidad del software en entorno AS / 400. Identificaron los elementos clave del método de gestión de calidad, como la satisfacción del cliente, calidad del producto, mejora continua del proceso y gente. Con base en datos empíricos, el progreso en varios aspectos de calidad y los parámetros del sistema de software AS / 400 fueron examinados. Presentaron un mapa de acción de calidad que describe varias acciones de calidad que se implementaron. Bakker y Hirdes (2015 citado por Ladányi, et al (2015)) describieron algunas experiencias de proyectos utilizando métricas de productos de software. Analizaron más de 10 millones de líneas de código en COBOL, PL / I, Pascal, C, C ++ y RPG (aproximadamente 1,8 millones de líneas de código).

Los objetivos de los proyectos escritos en RPG fueron mantenimiento y análisis de riesgos. Ellos encontraron que los problemas en los sistemas heredados tienen su causa en el diseño, no en la estructura del código. Además, El rediseño y la reestructuración de los sistemas existentes son soluciones menos costosas y más seguras a estos problemas que la reconstrucción. A veces, los sistemas heredados de RPG pueden ser difíciles de mantener, mejorar y expandir, ya que hay una falta general de comprensión de los sistemas. Suntiparakoo y Limpiyakorn presentaron un método de extracción de conocimiento de código RPG legado y su diagrama de flujo. El diagrama de flujo extraído puede servir como elemento de

aseguramiento de la calidad que facilita la comprensión del código heredado durante un proceso de mantenimiento de software.

Ladányi, et al (2015) comentan que en cuanto al modelo de calidad, se puede ver que el conjunto de estudios previos en RPG del mantenimiento del código y la garantía de calidad es muy limitado, por lo tanto, se desea un método de aseguramiento de calidad bien definido para manejar preguntas de esfuerzo de mantenimiento. Una vez que tenemos indicadores de calidad, como métricas de software que son capaces de caracterizar la calidad del software desde varios puntos de vista, es necesario estandarizar esta información de alguna manera. Esta es la razón por la cual los estándares internacionales ISO / IEC 25010, y su antecesor, el ISO / IEC 9126 han sido creados.

La comunidad de investigación reaccionó rápidamente ante la aparición de estos estándares, y varios artículos han sido publicados en relación con ellos. Jung y Kim (2019 citado por Ladányi, et al (2015)) validaron la estructura del estándar ISO / IEC 9126 basado en las respuestas de una encuesta generalizada. Ellos se centraron en las conexiones entre las subcaracterísticas y las características. Agruparon subcaracterísticas juntas basadas en la alta correlación en sus valores de acuerdo con los evaluadores, los autores encontraron que la mayoría de los grupos involucrados se relacionaban a una característica definida por el estándar.

Dado que los estándares no proporcionan detalles sobre cómo estas características se pueden determinar, numerosas adaptaciones y varias soluciones se han desarrollado para calcular y la evaluación de los valores de las propiedades definidas en el estándar. Zou y Kontogiannis introdujeron un enfoque de transformación del sistema orientado a requerimientos. Ellos migraron sistemas procedurales heredados a plataformas modernas orientadas a objetos mientras mantenían requisitos de calidad específicos para el sistema objetivo usando el enfoque de modelo de Markov y el algoritmo de Viterbi. Utilizaron gráficos de dependencia de objetivos flexibles para describir su modelo de mantenimiento y calidad de rendimiento para estimar la calidad del sistema objetivo. Al contrario de ellos nuestro objetivo no era mejorar el resultado de un proceso de migración, sino para proporcionar técnicas de control de calidad de vanguardia para sistemas RPG

tradicionales.

Aunque el examen de las dependencias de calidad entre la versión original y migrada de los sistemas RPG heredados es una parte del trabajo futuro. Otros usan benchmarking y modelos de calidad para proporcionar información de nivel superior sobre un sistema de software. El modelo tiene una fase de calibración para ajustar los valores umbral del modelo de calidad de tal manera que para cada nivel más bajo de atributo de calidad obtienen una distribución simétrica deseada.

Ladányi, et al (2015) usaron el (5; 30; 30; 30; 5) distribuciones porcentuales más de 5 niveles de calidad. Para convertir las métricas del software en índices de calidad también utilizaron un punto de referencia con una gran cantidad de evaluaciones, pero lo aplicaron de una manera diferente. Durante la calibración, en lugar de calcular los valores de umbral usan para aproximar una función de distribución normal llamada punto de referencia característica, que se utiliza para determinar la bondad del software.

En cuanto al lenguaje RPG, se enfocaron en los componentes esenciales que usaron para definir el modelo de calidad para el lenguaje de programación RPG. Con los años, el lenguaje RPG ha evolucionado en muchos aspectos. RPG III (lanzado en 1978) ya tenía subrutinas proporcionadas, construcciones estructuradas modernas tales como do, dow, if. Se realizó un gran avance en 1994, cuando se lanzó la primera versión de RPG IV. Con el lanzamiento de RPG IV, los desarrolladores obtuvieron nuevas características como bloques de forma libre o especificaciones de definición, y el conjunto de las operaciones disponibles se extendió también.

RPG apoya diferentes grupos de tipos de datos, a saber, tipos de datos de caracteres (Carácter, indicador, gráfico, UCS-2), tipos de datos numéricos (Formato binario, Formato flotante, Formato entero, Empaquetado-Decimal Formato, formato sin signo, formato decimal por zonas), fecha, hora y tipos de datos de marca de tiempo, tipo de datos de objeto (el usuario puede definir Objetos Java), y uno puede definir punteros a variables, procedimientos y programas. En RPG IV hay dos tipos de funciones. El primero se llama subrutina que no requiere parámetro y tiene una visibilidad y usabilidad limitadas. Procedimientos son construcciones más flexibles

ya que pueden tener parámetros y pueden ser llamados desde otros programas.

El fragmento de código presenta algunas características y capacidades de lenguaje RPG. Desde el principio, RPG usa el formato dependiente de la columna, de esta manera la programación requiere los elementos de código que se colocarán en columnas particulares. Por ejemplo, el personaje en la sexta columna significa la especificación tipo. Desde el RPG IV, es posible usar elSourceMeter para RPG. Un elemento central en el aseguramiento de la calidad del código fuente es un conjunto de herramientas que proporciona información útil sobre un sistema de diferentes puntos de vista. En su estudio, el análisis del código estático se realizó por una herramienta llamada SourceMeter para RPG/IV.

Ladányi, et al (2015) desde su enfoque en esto el papel no está en análisis estático, presentaron las capacidades de la herramienta solo brevemente. Usando la herramienta, se puede calcular diferentes métricas para elementos del código fuente ubicados en archivos fuente RPG, a saber, subrutinas, procedimientos, programas y para todo el sistema. Las métricas están organizadas en cuatro grupos para indicar el rol y características de ellos. Estos grupos se enumeran en el siguiente (algunos miembros del grupo también se presentan entre paréntesis): tamaño (LOC, LLOC), documentación (CLOC, CD), acoplamiento (NII, NOI), complejidad (NLE, McCC). El modelo de calidad que se presentará en la Sección III-C, utiliza estas métricas de código fuente como indicadores de calidad de bajo nivel. La descripción de las métricas y la calidad del software utilizado las características se pueden encontrar en la Tabla I.

Ladányi, et al (2015) comentaron que SourceMeter también recopila información sobre violaciones de reglas encontradas en los archivos fuente RPG. Las violaciones a las reglas no resultan en errores inmediatos, pero indican las ubicaciones en el código fuente donde posiblemente existen problemas más profundos. SourceMeter contiene una amplia gama de reglas definidas para el lenguaje RPG. Las reglas no se clasifican como métricas, sino que se clasifican en tres grupos según su criticidad. Estos tres grupos son los siguientes (con una muestra dada): advertencia P1 (operación prohibida utilizada), advertencia P2

(Copia demasiado profunda / Incluir anidación), advertencia P3 (deben evitarse las operaciones de depuración).

Tabla 1

Descripción de nodos en el modelo de calidad RPG

Sensor nodes	
CC	Clone coverage. The percentage of copied and pasted source code parts, computed for the Subroutine, procedure, program, and the system itself. CLOC Comment Lines of Code. Number of comment and documentation code lines of the subroutine/Procedure/program. Subroutines' CLOC are CLOC. Similarly, subroutines' and procedures' CLOC are not added to the containing Program's CLOC.
CD	Comment Density. The ratio of comment lines compared to the sum of its comment and Logical lines of code (CLOC and LLOC). Calculated for subroutines, procedures, and Programs.
LLOC	Logical Lines of Code. Number of code lines of the subroutine/procedure/program, Excluding empty and comment lines. In case of a procedure, the contained subroutines' is not counted. Similarly, in case of a program, the contained subroutines' and Procedures' LLOC is not counted.
McCC	McCabe's Cyclomatic Complexity [28] of the subroutine/procedure. The number of Decisions within the specified subroutine/procedure plus 1, where each if, else-if, for, Do, do-while, do-until, when, on-error counts once. Subroutines' McCC are not added to The containing procedure's McCC.
NLE	Nesting Level Else-If. Complexity of the subroutine/procedure/program expressed as the Depth of the maximum embeddedness of its conditional and iteration block scopes, where In the if-else-if construct only the first if instruction is considered. The following RPGLanguage items are taken into consideration: if, for, do, do-while, do-until, select, and Monitor. The else-if, else, when, other, and on-error operations do not increase the value Of the metric; however, they can contain elements defined above, which increase NLE. Subroutines' NLE are not added to the containing procedure's NLE. Similarly, subroutines 'And procedures' NLE are not added to the containing program's NLE.
NII	Number of Incoming Invocations. Number of other subroutines which directly call the Subroutine.
NOI	Number of Outgoing Invocations. Number of other subroutines which are directly called By the subroutine.
Warning P1	The number of critical rule violations in the subroutine/procedure/program. They can be Potential root causes of system faults.
Warning P2	The number of major rule violations in the subroutine/procedure/program. Serious coding Issues which makes the code hard to understand, and can decrease efficiency.
Warning P3	The number of minor rule violations in the subroutine/procedure/program. These are only Minor coding style issues, makes the source code harder to comprehend. Aggregated nodes
Changeability	The capability of the software product to enable a specified modification to be implemented, Where implementation includes coding, designing and documenting changes.
Complexity	Represents the overall complexity of the source code. It is represented by the McCabe's Cyclomatic Complexity and the Nested level of the subroutines.
Comprehensibility	Expresses how easy it is to understand the source code. It involves the complexity, Documentation and size of the source code.
Documentation	Expresses how well the source code is documented. It is represented by the density and The amount comment lines of code in a subroutine.

Fault proneness	Represents the possibility of having a faulty code segment. Represented by the number of Minor, major and critical rule violations.
Stability	The capability of the software product to avoid unexpected effects from modifications of The software.
Aggregated nodes	defined by the ISO/IEC 25010 standard
Maintainability	Degree of effectiveness and efficiency with which a product or system can be modified to Improve it, correct it or adapt it to changes in environment, and in requirements.
Analyzability	Degree of effectiveness and efficiency with which it is possible to assess the impact on A product or system of an intended change to one or more of its parts, or to diagnose a Product for deficiencies or causes of failures, or to identify parts to be modified.
Modifiability	Degree to which a product or system can be effectively and efficiently modified without Introducing defects or degrading existing product quality.
Testability	Degree of effectiveness and efficiency with which test criteria can be established for a System, product or component and tests can be performed to determine whether those Criteria have been met.
Reusability	The degree to which an asset can be used in more than one system, or in building other Assets.

Nota. Tomado de Ladányi G, Tóth Z, Ferenc R and Keresztesi T. (2015) "A software quality model for RPG". IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Montreal. 91-100.doi: 10.1109/SANER.2015.7081819

Estos grupos se basan en las prioridades. El grupo de advertencias P1 representa algo crítico, la advertencia P2 algo mayor y Advertencia P3 significa infracciones menores de las reglas. Uno puede ver que usando una operación prohibida (definida por los estándares actuales de la compañía) es más no deseado que un mensaje de depuración. Para cada elemento de código (subrutina, procedimiento, programa, sistema), se calculan los valores resumidos para indicar la cantidad de violaciones a las reglas ubicadas en ese segmento de código apropiado para cada regla. El modelo de calidad usa la cantidad de violaciones a las reglas de acuerdo con sus prioridades.

Ladányi, et al (2015) comentaron que el tercer componente de SourceMeter es responsable de detectar los Clones del código de software, también conocido como duplicaciones de código. Un clon de código es un fragmento de código fuente que ocurre más de una vez en un sistema de software dado. Clones generalmente no son deseables, debido a sus posibles efectos secundarios. Un efecto dañino ocurre cuando un código tiene errores en él y es copiado, Además, las instancias de código duplicadas son difíciles de actualizar y mantener SourceMeter puede detectar diferentes tipos de partes de código duplicadas en el código fuente, y también organizar el código encontrado se divide en grupos de clones (clase de clonación - CCL).

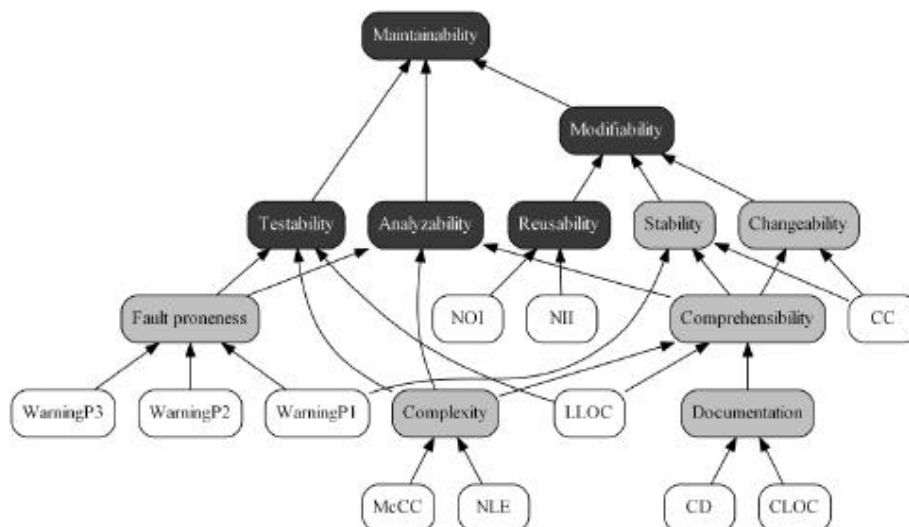


Figura 1. ADG Modelo de calidad RPG. Obtenido de Ladányi, et al (2015).

La cantidad de código clonado es una medida muy importante para tener en consideración si queremos estimar el esfuerzo de mantener un sistema. SourceMeter para RPG proporciona diferentes métricas de clonación para el sistema de software analizado. Algunos de ellos están listados abajo: Clase de clonación (CCL): cantidad de clases de clonación. Clone Instance (CI): cantidad de instancias de clonación. Cobertura Clon (CC) - porcentaje de copiado y partes del código fuente pegadas, calculadas para la subrutina, procedimiento, programa y el sistema mismo. Clone Age (CA) - número de analizadas previamente revisiones en las que se presentó el clon. Complejidad del clon (CCO): complejidad de McCC para instancia de clonación Para las clases de clonación, CCO es la suma de CCO de cada IC en la clase de clonación.

En cuanto al método de calificación, se brinda una breve descripción del modelo de calidad de software probabilístico llamado ColumbusQM y mostraron cómo implementaron el enfoque general para el Lenguaje RPG. El estándar internacional ISO / IEC 25010 define las características de calidad del producto que son ampliamente aceptadas por expertos industriales e investigadores académicos. Estas características son: idoneidad funcional, eficiencia del rendimiento, compatibilidad, usabilidad, confiabilidad, seguridad, mantenibilidad y portabilidad.

En este estudio, se centro en la mantenibilidad porque tiene conexión obvia y directa con los costos de alterar el comportamiento del software. El estándar define las subcaracterísticas de mantenibilidad también, pero no proporciona más detalles sobre cómo se puede calcular estas características y subcaracterísticas.

Según Ladányi, et al (2015) la idea básica detrás de ColumbusQM es dividir el problema complejo de cálculo de una característica de calidad de alto nivel en sub-problemas menos complejos. En el modelo de calidad, se pueden describir con un gráfico dirigido acíclico, llamado el gráfico de dependencia de atributo (ADG), las relaciones entre las métricas de nivel inferior que pueden ser fácilmente obtenidas del código fuente y las características calidad de mayor nivel. El modelo desarrollado ADG para lenguaje RPG se muestra en la Figura 1.

Los nodos negros en el modelo están definidos por norma internacional ISO / IEC 25010, los nodos blancos son las métricas de código fuente calculadas por SourceMeter para la herramienta RPG, y finalmente, los nodos grises son los nodos internos definidos por el autor para ayudar a revelar las dependencias en el modelo. Llamaron en el código fuente nodos métricos como nodos sensor y el otro nodo como nodos agregados. La Tabla I contiene las descripciones de todos los nodos, Aunque la estructura básica del ADG se basa en un modelo de calidad de Java anterior, las diferencias entre los idiomas causaron modificaciones.

La diferencia más importante es que no se podía aplicar métricas orientadas a objetos en el modelo de calidad RPG porque es un lenguaje de programación procedural. Una parte esencial del enfoque además del modelo de calidad es el punto de referencia, que contiene varios sistemas RPG. Al comparar el sistema con las aplicaciones del punto de referencia, el enfoque convierte las métricas de código fuente de bajo nivel en índices de calidad. Formalmente, cada métrica de código fuente puede considerarse como una variable aleatoria que puede tomar valores reales con valores de probabilidad particulares. Para dos diferentes sistemas de software, $h_1(t)$ y $h_2(t)$ son las funciones de densidad de probabilidad correspondientes a la misma métrica. Ahora el valor de bondad de un sistema con respecto al otro, es definido como:

$$D(h_1, h_2) = \int_{-\infty}^{\infty} (h_1(t) - h_2(t))\omega(t)dt$$

Dónde $\omega(t)$ es la función de peso que determina la noción de la bondad La figura 3 nos ayuda a comprender el significado de la fórmula: calcula el área firmada entre las dos funciones ponderado por la función $\omega(t)$.

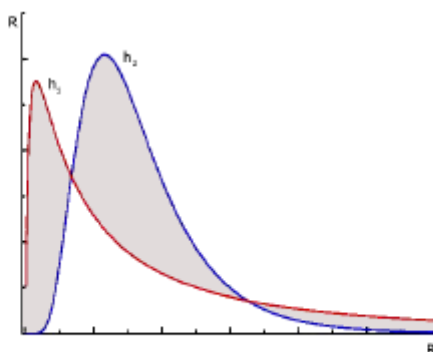


Figura2. Comparación de funciones de densidad de probabilidad. Obtenido de Ladányi, et al (2015).

Para los lados del ADG, se preparó una encuesta. En esta encuesta a los desarrolladores se les pidió que asignen pesos a los bordes, en función de cómo se sintieron sobre la importancia de la dependencia. En consecuencia, una variable

aleatoria multidimensional $\vec{Y}_v = (Y_v^1, Y_v^2, \dots, Y_v^n)$

Corresponderá a cada nodo de nivel superior v . Clásicamente se toma, una combinación lineal de valores de bondad y pesos, y se asigna al nodo de nivel superior. Cuando se trata de probabilidades, uno debe tomar cada posible combinación de valores y pesos de bondad, y también las probabilidades de su resultado en cuenta. Nosotros definimos la función de bondad agregada para el nodo v en el siguiente camino:

$$g_v(t) = \int_{t=q^r} \int_{\substack{\vec{q}=(q_1, \dots, q_n) \in \Delta^{n-1} \\ \vec{r}=(r_1, \dots, r_n) \in C^n}} \vec{f}_{Y_v}(\vec{q}) g_1(r_1) \dots g_n(r_n) d\vec{r} d\vec{q},$$

Donde $\vec{f}_{\vec{Y}_v}(\vec{q})$ es la función de densidad de probabilidad de \vec{Y}_v , g_1, g_2, \dots, g_n son las funciones de bondad correspondientes a nodos entrantes, Δ^{n-1} es el $(n-1)$ estándar simplex en \mathbb{R}^n y C^n es la unidad estándar n-cube en \mathbb{R}^n . Aunque la fórmula puede parecer aterradora a primera vista, es solo una generalización de cómo se realiza la agregación en los enfoques clásicos. Clásicamente, una combinación lineal de los valores y pesos de bondad se toma, y se asigna al nodo agregado. Cuando se trata de probabilidades, uno necesita tomar todas las combinaciones posibles de valores de bondad y pesos, y también las probabilidades de su resultado en cuenta.

En cuanto a la integración con QualityGate dado que la intención era proporcionar información valiosa para los gerentes y desarrolladores sobre una base diaria que era necesario para integrar nuestro enfoque en una herramienta de administración de calidad continua. La herramienta SourceMeter for RPG y el enfoque de ColumbusQM se integró en una herramienta llamada QualityGate. Como una plataforma de gestión integral de la calidad del software, QualityGate es capaz de calcular los valores de la calidad utilizando el ColumbusQM del código fuente, usando un amplio rango de métricas de calidad de software proporcionadas por el SourceMeter para RPG. Está potenciado por un modelo de calidad incorporado conforme al estándar ISO / IEC 25010 y tiene un punto de referencia que contiene datos de análisis de una gran cantidad de sistemas de RPG. Esto hace posible calcular atributos objetivos de calidad y estimar los próximos costos de desarrollo.

Historia de la calidad del software

Tabla2.

Evolution of Software Quality Models in Context of the Standard ISO 25010

No	Modelo de calidad	Año de publicación	Autor
1	McCall	1977	Jhon McCall
2	Bohem	1978	Boehm
3	Carlo Ghezzi	1991	Carlo Ghezzi
4	FURPS	1992	Grady R. & Hewlett Packard
5	IEEE	1993	IEEE
6	Roger Pressman	1982	Roger Pressman
7	Dromey	1995	Dromey
8	ISO 9126-1	2001	ISO
9	QMOOD	2002	Bansiya
10	ISO 25010	2010	ISO

Nota: Tomado de GordieievOleksandr (2014).

La tabla 2 muestra la evolución de los modelos de calidad de software (SW) (QM) según GordieievOleksandr (2014). Sobre los últimos cuarenta años, el primer software QM de McCall se usa para el modelo presentado en la norma ISO 25010. Se ven 9 modelos para el análisis y se divide en conjuntos de QM básicos y corporativos de acuerdo con la integridad, detalle e importancia. Se ven los modelos básicos McCall (1977), IEEE 1219 (1993), ISO 9126-1 (2001), ISO 25010 (2010). En 1982 Pressman presenta sus libros sobre ingeniería de software y la calidad del software. La estructura QM se describe por jerarquía cuyos elementos son conjuntos de características (subcaracterísticas) y relaciones de subordinación entre ellos. Para evaluar la complejidad e integridad de SW QM y compararlos con los últimos se introducen las métricas especiales y generales especiales del modelo

ISO 25010. Dependencia analítica del crecimiento de la complejidad del modelo representado por un linealse obtiene la función Análisis de la evolución de algunas características (operacional idoneidad, efectividad, confiabilidad, usabilidad, seguridad, etc.). Palabras clave: análisis evolutivo, modelos de calidad de software, complejidad.

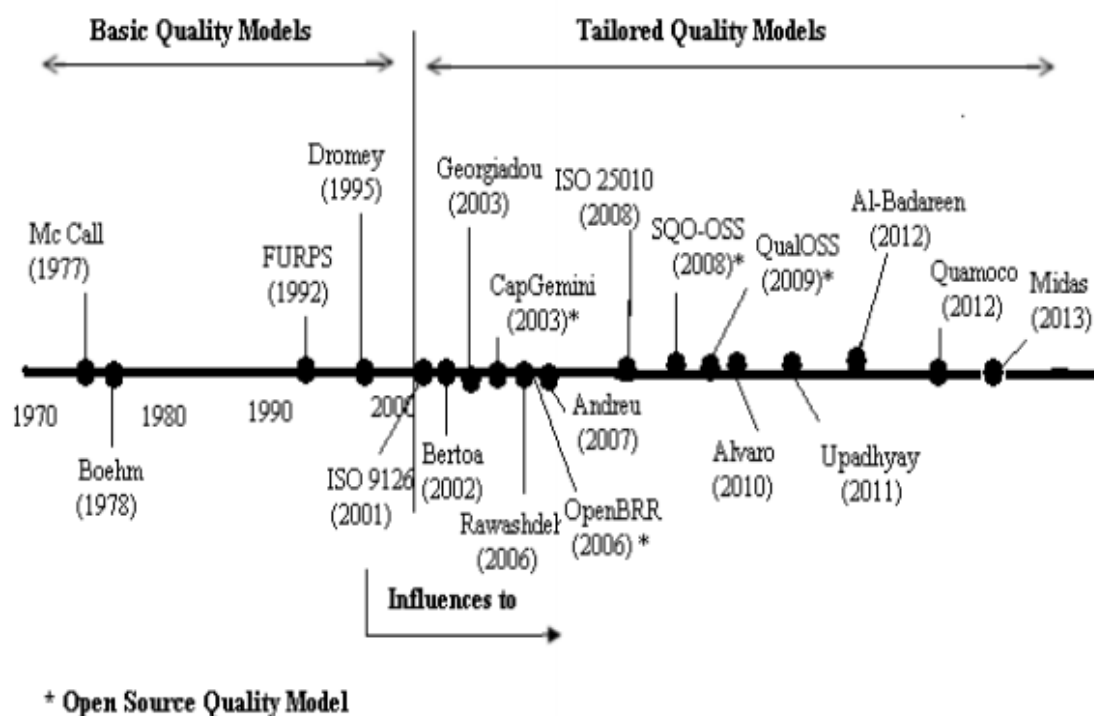


Figura 3. Perspectiva histórica de los modelos de calidad de software. José P. Miguel (2014)

José P. Miguel, David Mauricio and Glen Rodríguez (2014) explica cómo los modelos básicos o iniciales son de estructura jerárquica; se pueden ajustar a cualquier tipo de software y están orientados a la evaluación y mejora. Los seis más importantes son: Mc Call et al (1977), Boehm et al (1978), modelo FURPS (1992), modelo Dromey (1995), modelo ISO 9126-1 (2001) y sus estándares para ambas medidas externas: ISO / IEC 9126-2 (2003), métricas internas: ISO / CEI 9126-3 (2003) y calidad en uso: ISO / IEC 9126-4 (2004). El modelo ISO -9126 recibió entradas de modelos y estándares anteriores para evaluar la calidad del software. En 2007 se estableció una actualización como ISO Modelo 25010: ISO / IEC CD 25010 (2008). La ISO 25010 en realidad se conoce como SQuaRE (Ingeniería de software - Requisitos y evaluación de calidad de los productos de software). Los modelos de calidad personalizados comenzaron a aparecer el año

2001 con el modelo Bertoa (2002), seguido de Modelo de Georgiadou (2003), Modelo de Alvaro (2005), Modelo de Rawashdesh (2006). La característica principal es que son específicos de un dominio particular de aplicación y la importancia de las características puede ser variable en relación con un modelo general. Estos modelos surgen de la necesidad de las organizaciones y la industria del software de modelos de calidad específicos capaces de hacer evaluación especializada en componentes individuales. Están contruidos a partir de los modelos básicos especialmente el ISO 9126, con la adición o modificación de subfactores y el objetivo de cumplir necesidades de dominios específicos o aplicaciones especializadas.

En los últimos años, la construcción del software se ha centrado en la reutilización y el desarrollo de software basado en componentes (CBSD). Como una consecuencia, el éxito de un producto depende en gran medida de la calidad de los componentes. Otros autores clasifican los modelos según las características del usuario. Por ejemplo Klas (2011) distingue tres categorías de modelos que corresponden a: 1) el nivel de uso público general o dominio específico, 2) nivel organizacional que se enfoca en satisfacer los intereses de un organización, y 3) el proyecto de nivel que se aplica a un proyecto específico para garantizar la calidad. Debido a la importancia de los componentes de COTS, Ayala (2010) establece un proceso para seleccionar el software. Se basó en observaciones y entrevistas con desarrolladores de componentes.

Teoría de las Técnicas de revisión y aseguramiento de calidad de software de Pressman

Pressman (2010) explico cómo las revisiones del software son un “filtro” Para el proceso del software. Es decir, se aplican en varios puntos durante la ingeniería de software y sirven para descubrir errores y defectos a fin de poder eliminarlos. Las revisiones del software “purifican” los productos del trabajo de la ingeniería de software, incluso los modelos de requerimientos y diseño, código y datos de prueba. Como citan Freedman y Weinberg, analizan del modo siguiente la necesidad de hacer revisiones: El trabajo técnico necesita las revisiones por la

misma razón que los lápices necesitan borradores: errar es humano. La segunda razón por la que son necesarias las revisiones técnicas es porque, si bien las personas son buenas para detectar algunos de sus propios errores, muchas clases de ellos pasan desapercibidos con más facilidad para quien los comete que para otras personas.

Una revisión cualquiera es una forma de utilizar la diversidad de un grupo para lo siguiente: (1) Resaltar las mejoras necesarias en el producto que elaboró una sola persona o equipo; (2) confirme aquellas partes de un producto en las que no se desea o no se necesita hacer una mejora; (3) realice el trabajo técnico de calidad más uniforme, o al menos más predecible, que pueda lograrse sin hacer revisiones, a fin de que el trabajo técnico sea más manejable. Como parte de la ingeniería de software, pueden realizarse muchos diferentes tipos de revisiones. Cada uno tiene su lugar. Una reunión informal alrededor de la máquina del café es una forma de revisión si se analizan problemas técnicos.

Pressman (2010) dice que la presentación formal de la arquitectura del software a un público de clientes, administradores y técnicos también es una forma de revisión. Sin embargo, en su libro se centra en las revisiones técnicas o por pares, ejemplificadas por las revisiones casuales, walkthroughs e inspecciones. Desde el punto de vista del control de calidad, una revisión técnica (RT) es el filtro más eficaz. Realizado por ingenieros de software (y de otro tipo) para ingenieros de software, la RT es un medio eficaz para detectar errores y mejorar la calidad.

Según Pressman (2010) en el contexto del proceso del software, los términos defecto y falla son sinónimos. Los dos implican un problema de calidad descubierta después de haberse liberado el software a los usuarios finales (o a otra actividad estructural del proceso del software). Anteriormente se empleó el término error para denotar un problema de calidad descubierto por ingenieros de software (o de otra clase) antes de entregar el software al usuario final (o a alguna actividad estructural del proceso del software). El objetivo principal de las revisiones técnicas es encontrar errores durante el proceso a fin de que no se conviertan en defectos después de liberar el software. El beneficio obvio de las revisiones técnicas es el descubrimiento temprano de los errores, de modo que no se propaguen a la

siguiente etapa del proceso del software. Varios estudios de la industria indican que las actividades de diseño introducen de 50 a 65 por ciento de todos los errores (y en realidad de todos los defectos) durante el proceso del software. Sin embargo, las técnicas de revisión han demostrado tener una eficacia de hasta 75 por ciento para descubrir fallas del diseño.

Al detectar y eliminar un gran porcentaje de estos errores, el proceso de revisión reduce de manera sustancial el costo de las actividades posteriores en el proceso del software. Para ilustrar la generación y detección de errores durante las acciones de diseño y generación de código de un proceso de software, puede usarse un modelo de amplificación del defecto. En ciertos casos, los errores de etapas anteriores ignorados son amplificados (en un factor x de amplificación) por el trabajo en curso. El número de errores detectados durante cada una de las etapas citadas se multiplica por el costo que implica eliminar un error (1.5 unidades de costo para el diseño, 6.5 unidades de costo antes de las pruebas, 15 unidades de costo durante las pruebas y 67 unidades de costo después de la entrega).² Con estos datos, el costo total del desarrollo y mantenimiento cuando se efectúan revisiones es de 783 unidades de costo.

Según Pressman (2010) cuando no se hacen revisiones, el costo total es de 2177 unidades, casi tres veces más caro. Debe dedicarse tiempo y esfuerzo a la realización de revisiones y su organización de desarrollo debe destinar el dinero para ello. Sin embargo, los resultados del ejemplo anterior dejan pocas dudas acerca de lo que puede pagar ahora o de que después deberá pagar mucho más.

Las revisiones técnicas son una de las muchas acciones que se requieren como parte de las buenas prácticas de la ingeniería de software. Cada acción requiere un esfuerzo humano dirigido. Como el esfuerzo disponible para el proyecto es finito, es importante que una organización de software comprenda la eficacia de cada acción, definiendo un conjunto de métricas que puedan utilizarse para evaluar esa eficacia. Aunque se han definido muchas métricas para las revisiones técnicas, un conjunto relativamente pequeño da una perspectiva útil.

Las siguientes métricas para la revisión pueden obtenerse conforme se efectúe ésta: (1) Esfuerzo de preparación, E_p : esfuerzo (en horas-hombre) requerido para revisar un producto del trabajo antes de la reunión de revisión real. (2) Esfuerzo de evaluación, E_a : esfuerzo requerido (en horas-hombre) que se dedica a la revisión real. (3) Esfuerzo de la repetición, E_r : esfuerzo (en horas-hombre) que se dedica a la corrección de los errores descubiertos durante la revisión, (4) Tamaño del producto del trabajo, TPT: medición del tamaño del producto del trabajo que se ha revisado (por ejemplo, número de modelos UML o número de páginas de documento o de líneas de código)., (5) Errores menores detectados, Errores menores: número de errores detectados que pueden clasificarse como menores (requieren menos de algún esfuerzo especificado para corregirse), (6) Errores mayores detectados, Errores mayores: número de errores encontrados que pueden clasificarse como mayores (requieren más que algún esfuerzo especificado para corregirse). Estas métricas pueden mejorarse, asociando el tipo de producto del trabajo que se revisó con las métricas obtenidas.

Análisis de las métricas. Antes de comenzar el análisis deben hacerse algunos cálculos sencillos. El esfuerzo total de revisión y el número total de errores descubiertos se definen como sigue:

$$E_{\text{Revisión}} = E_p + E_a + E_r$$

$$Err_{\text{Tot}} = Err_{\text{menores}} + Err_{\text{mayores}}$$

La densidad del error representa los errores encontrados por unidad de producto del trabajo revisada.

$$\text{Densidad del error} = Err_{\text{tot}} / TPT$$

Por ejemplo, si se revisa un modelo de requerimientos con objeto de encontrar errores, inconsistencias y omisiones, es posible calcular la densidad del error en varias formas diferentes. El modelo de requerimientos contiene 18 diagramas UML como parte de 32 páginas de materiales descriptivos. La revisión

detecta 18 errores menores y 4 mayores. Por tanto, $Err_{tot} = 22$. La densidad del error es 1.2 errores por diagrama UML o 0.68 errores por página del modelo de requerimientos.

Teoría Modelo de Calidad del Software ISO/IEC 2510

El modelo de calidad es la parte principal de un sistema de evaluación de la calidad del producto. El modelo de calidad determina qué características de calidad se tendrán en cuenta al evaluar las propiedades de un producto de software. La calidad de un sistema es el grado en que el sistema satisface las necesidades declaradas e implícitas de sus diferentes partes interesadas, y por lo tanto proporciona valor. Las necesidades de esos grupos de interés (funcionalidad, rendimiento, seguridad, mantenibilidad, etc.) son precisamente lo que se representa en el modelo de calidad, que clasifica la calidad del producto en características y subcaracterísticas.

El modelo de calidad del producto definido en ISO / IEC 25010 comprende las ocho características de calidad que se muestran en la siguiente figura



Figura4. Modelo ISO / IEC 25010.

Idoneidad funcional. Esta característica representa el grado en que un producto o sistema proporciona funciones que cumplen con las necesidades explícitas e implícitas cuando se usan bajo condiciones específicas. Esta característica se compone de las siguientes subcaracterísticas: Compleción funcional. Grado en el que el conjunto de funciones cubre todas las tareas especificadas y los objetivos del usuario. Corrección funcional Grado en el que un producto o sistema

proporciona los resultados correctos con el grado de precisión necesario.
Adecuación funcional.

Grado en que las funciones facilitan la realización de tareas y objetivos Específicos. Eficiencia en el desempeño. Esta característica representa el rendimiento relativo a la cantidad de recursos utilizados en las condiciones establecidas. Esta característica se compone de las siguientes subcaracterísticas: Comportamiento del tiempo. Grado en el que los tiempos de respuesta y Procesamiento y las tasas de rendimiento de un producto o sistema, al realizar sus funciones, cumplen los requisitos. Utilización de recursos. Grado en el cual los montos y tipos de recursos utilizados por un producto o sistema, al realizar sus funciones, cumplen con los requisitos. Capacidad. Grado en el que los límites máximos de un producto o parámetro del sistema cumplen los requisitos.

Compatibilidad. Grado en el cual un producto, sistema o componente Puede intercambiar información con otros productos, sistemas o componentes, y / o realizar las funciones requeridas, mientras comparte el mismo entorno de hardware o software. Esta característica se compone de las siguientes subcaracterísticas: Coexistencia: grado en el cual un producto puede realizar sus funciones requeridas de manera eficiente mientras comparte un entorno y recursos comunes con otros productos, sin impacto perjudicial en ningún otro producto. Interoperabilidad: grado en el cual dos o más sistemas, productos o componentes pueden intercambiar información y usar la información que se ha intercambiado.

Usabilidad. Grado en el cual un producto o sistema puede ser utilizado por usuarios específicos para alcanzar objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso específico. Esta característica se compone de las siguientes subcaracterísticas: Idoneidad. Reconocimiento. Grado en el cual los usuarios pueden reconocer si un producto o sistema es apropiado para sus necesidades. Learnability. Grado en que un producto o sistema puede ser utilizado por usuarios específicos para lograr objetivos específicos de aprender a utilizar el producto o sistema con efectividad, eficiencia, ausencia de riesgo y satisfacción en un contexto de uso específico.

Operabilidad. Grado en el cual un producto o sistema tiene atributos que hacen que sea fácil de operar y controlar. Protección contra errores de usuario. Grado en el que un sistema protege a los usuarios contra errores. Estética de la interfaz de usuario. Grado en el cual una interfaz de usuario permite una interacción agradable y satisfactoria para el usuario. Accesibilidad. Grado al cual un producto o sistema puede ser utilizado por personas con la más amplia gama de características y capacidades para alcanzar un objetivo específico en un contexto de uso específico.

Confiabilidad. Grado en el que un sistema, producto o componente realiza funciones especificadas en condiciones específicas durante un período de tiempo específico. Esta característica se compone de las siguientes subcaracterísticas: Madurez. El grado en que un sistema, producto o componente satisface necesidades de confiabilidad bajo operación normal. Disponibilidad. Grado en el cual un sistema, producto o componente es operacional y accesible cuando se requiere su uso. Tolerancia a fallos. Grado en el que un sistema, producto o componente funciona según lo previsto a pesar de la presencia de fallas de hardware o software. Recuperabilidad. Grado en el que, en caso de interrupción o falla, un producto o sistema puede recuperar los datos directamente afectados y restablecer el estado deseado del sistema.

Seguridad. Grado en que un producto o sistema protege la información y los datos para que las personas u otros productos o sistemas tengan el grado de acceso a los datos apropiado para sus tipos y niveles de autorización. Esta característica se compone de las siguientes subcaracterísticas: Confidencialidad. grado en el que un producto o sistema garantiza que los datos solo sean accesibles para los autorizados a acceder. Integridad. Grado en el que un sistema, producto o componente impide el acceso no autorizado o la modificación de programas o datos de computadora. No repudio grado en que se puede demostrar que las acciones o eventos tuvieron lugar, para que los eventos o acciones no puedan ser repudiados más tarde.

Responsabilidad. Grado en el que las acciones de una entidad se pueden rastrear de manera única a la entidad. Autenticidad. Grado en el cual se puede probar que la identidad de un sujeto o recurso es la reclamada.

Mantenibilidad. Esta característica representa el grado de efectividad y eficiencia con la que un producto o sistema puede ser modificado para mejorarlo, corregirlo o adaptarlo a los cambios en el entorno y en los requisitos. Esta característica se compone de las siguientes subcaracterísticas:

Modularidad. Grado en el cual un sistema o programa de computadora se compone de componentes discretos tales que un cambio en un componente tiene un impacto mínimo en otros componentes. Reusabilidad. Grado en el cual un activo puede ser utilizado en más de un sistema, o en la construcción de otros activos. Analizabilidad. Grado de efectividad y eficiencia con el cual es posible evaluar el impacto en un producto o sistema de un cambio intencional a una o más de sus partes, o diagnosticar un producto por deficiencias o causas de fallas, o identificar partes a ser modificadas.

Modificabilidad. Grado en que un producto o sistema puede ser modificado de manera efectiva y eficiente sin introducir defectos o degradar la calidad del producto existente. Testabilidad. Grado de eficacia y eficiencia con el que se puede establecer los criterios de prueba para un sistema, producto o componente y se pueden realizar pruebas para determinar si se han cumplido esos criterios. Portabilidad. Grado de eficacia y eficiencia con el que un sistema, producto o componente puede transferirse de un hardware, software u otro entorno operacional o de uso a otro.

Esta característica se compone de las siguientes subcaracterísticas: Adaptabilidad. Grado en el cual un producto o sistema puede ser adaptado efectiva y eficientemente para hardware, software u otros entornos operacionales o de uso diferentes o en evolución. Instalabilidad. Grado de eficacia y eficiencia con el que un producto o sistema puede instalarse y / o desinstalarse con éxito en un entorno específico. Reemplazabilidad. Grado en el cual un producto puede reemplazar otro producto de software especificado para el mismo propósito en el mismo entorno.

SQUARE: Modelo actualizado de las características de calidad

Las características no funcionales de los sistemas de software tales como usabilidad, desempeño, seguridad, por mencionar algunas vuelven cada día más importantes. Los usuarios, al percibir las en unas aplicaciones, inconscientemente las esperan en otras. Los analistas saben que de por sí ya es difícil obtener con precisión los requerimientos funcionales, en el caso de los requerimientos no funcionales es todavía más complicado.

Hasta hace poco el estándar ISO/IEC 9126-1 Quality Model, del año 2001 fue la referencia básica de la clasificación de las características de calidad de software. Pero han pasado casi 10 años y la comunidad internacional decidió revisar este estándar para actualizarlo. De paso se cambió la numeración para abrir una nueva colección de estándares relacionados con la calidad de producto, la numeración de esta colección parte del 25000. El primer estándar de esta colección es el ISO/IEC FCD 25010: Systems and software engineering – System and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models. A continuación, se resumen elementos más importantes de este estándar.

Con calidad de software nos referimos al grado en que un producto de software satisface las necesidades expresas e implícitas cuando se usa bajo condiciones específicas. Estas necesidades expresas e implícitas están representadas por el modelo de calidad que categoriza las características de calidad.

El modelo de características de calidad se subdivide en dos vistas: calidad en uso y calidad de producto. La primera, identifica las características de calidad esperadas por un usuario del sistema y la segunda, las que debe cuidar el constructor del sistema.

Calidad en uso. Definida como grado en que un producto, utilizado por usuarios específicos, cumple con sus necesidades de lograr objetivos particulares

con efectividad, eficiencia, seguridad y satisfacción en un contexto específico de uso. La tabla 3 describe las características de calidad de uso.

Tabla 3.

Tabla de Características de calidad en uso

Característica	Subcaracterística	Descripción
Efectividad		Exactitud y completitud
Eficiencia		Recursos gastados en relación a la exactitud y completitud lograda
Satisfacción	cumplimiento de Propósito	Confianza
	Placer/Confort físico	Satisfacción de necesidades
Seguridad	Riesgo de daño En salud o seguridad	No debe estar en peligro la vida, salud, integridad o medioambiente
	Riesgo de daño al Medioambiente	
Contexto global	De uso	Grado de uso con efectividad/eficiencia, seguridad
	Completo/flexible	

Nota: Tomado de <https://sg.com.mx/content/view/990>.

Calidad de producto. Se define como grado en que un producto (sistema/software) cumple con las siguientes características: funcionalidad apropiada, fiabilidad, eficiencia en desempeño, usabilidad, seguridad, compatibilidad, mantenibilidad y portabilidad. La tabla 4 detalla dichas características.

Tabla 4.

Tabla Características de calidad en producto

Característica	Subcaracterística	Descripción
Funcionalidad	Apropiada	Completa, correcta , Adecuada
Fiabilidad		Madurez, Disponibilidad, Tolerancia a Fallas Recuperable
Eficiencia en Desempeño		Tiempo de respuesta utilización de Recursos
Usabilidad		Fácil de aprender Operable
Seguridad		Confiabilidad , Integridad
Compatibilidad		coexistencia
Mantenibilidad		Modularidad , Reusabilidad
Portabilidad		adaptable
		satisfacción de las necesidades expresas e implícitas cuando se usa bajo condiciones Especificas confianza en el sistema
		relativa a tamaño de recursos utilizado
		usuario reconoce si producto cumple Necesidades
		información y datos protegidos más de un sistema , Comparte hardware y sw
		eficacia y eficiencia en modificaciones
		Eficiencia para cambiar de Un ambiente a otro.

Nota: Obtenido de <https://sg.com.mx/content/view/990>.

A cada característica o sub-característica se asocian uno o más atributos de calidad que deben ser medibles. La definición de estos atributos y sus mediciones es actualmente el objeto de trabajo en la parte ISO/IEC 25021 de la colección SQuaRE. En cuanto a la relevancia, hasta hace poco la característica más importante de calidad de software fue la de funcionalidad apropiada. Para lograrla se han definido técnicas de captura y documentación de requerimientos funcionales, de las cuales los casos de uso son el mejor ejemplo. Sin embargo, las demás características, antes conocidas como no funcionales, han adquirido cada vez mayor importancia y son las que de manera implícita esperan los usuarios. Es por eso muy importante que los desarrolladores de software las conozcan y aprendan a identificarlas junto con sus clientes, plasmarlas en los diseños y probarlas antes de que el usuario sufra su ausencia.

Afortunadamente no todos los sistemas requieren cumplir con todas las características mencionadas o cubrirlas con el mismo grado de profundidad. Esta es otra tarea muy importante para los analistas y los diseñadores. Saber identificar el grado real requerido por el cliente del cumplimiento de las características de calidad puede ahorrarles el disgusto del cliente y mucho re-trabajo o, cuando el desarrollador es “más papista que el papa”, recurrir en una inversión de esfuerzo

que no es realmente requerida.

25010 define dos modelos: (1) Un modelo de calidad en uso compuesto de cinco características (algunas de las cuales se subdividen en subcaracterísticas) que se relacionan con el resultado de interacción cuando un producto se usa en un contexto particular de uso. Este el modelo del sistema es aplicable al sistema humano-computadora completo. • Un modelo de calidad del producto compuesto por ocho características (que son más subdividido en subcaracterísticas) que se relacionan con propiedades estáticas de software y propiedades dinámicas del sistema informático. (2) Las características definidas por ambos modelos son relevantes para todo el software productos y sistemas informáticos. Las características y subcaracterísticas que los caracterizan en los detalles proporcionan coherencia terminología para especificar, medir y evaluar sistema y software calidad del producto. También proporcionan un conjunto de características de calidad contra qué requisitos de calidad establecidos se pueden comparar para la integridad. (a) Modelos antiguos 9126 definidos internos, externos y en uso.

En cuanto a los límites, es poco probable que las características y subcaracterísticas estén siempre entre ellos perfectamente independiente, las características y subcaracterísticas son propiedades abstractas que están relacionadas con uno o más indicadores que se miden por métricas. Estas no siempre están en relación lineal con características que estiman, falta, en cualquier caso, el vínculo entre el modelo cualitativo y "cómo" desarrollar bien entonces software.

El alcance de los modelos excluye propiedades puramente funcionales, pero incluye la idoneidad funcional. Los destinatarios son: Usuarios, Desarrolladores / mantenedores, Cliente, Administradores del sistema, Clientes.

Ejemplos de uso: identificación de los requisitos de software y sistema , validar la exhaustividad de la definición de requisitos , identificación de los objetivos de diseño de software y sistema , identificación de objetivos de prueba de software y sistema , identificar los criterios de control de calidad como parte del aseguramiento de la calidad , establecer medidas de características de calidad en apoyo de estas

ocupaciones , identificar los criterios para la aceptación de productos , proporcionar un marco para la definición de la calidad del software en un documento contractual.

La calidad de uso es lo que el usuario percibe en el uso del producto en su contexto real de uso, en particular la capacidad de tal producto para apoyarlo con eficacia y eficiencia en su trabajo, exhibiendo una buena usabilidad. Este tipo de calidad es lo que, sobre todo, el desarrollador debe esforzarse por hacer. Las propiedades intrínsecas del producto (aquellas medidas directamente en el código fuente). Se obtiene comenzando de: requisitos del usuario, que son especificaciones de calidad y especificada por el usuario, siempre que primera entrada al diseño, y especificaciones técnicas, que representan el calidad requerida por el usuario traducida por el desarrollador en la arquitectura de software, programa estructura y la interfaz de usuario

Características del software: Eficacia, precisión e integridad con que el usuario logra objetivos específicos , Eficiencia, el esfuerzo en relación a la efectividad , Satisfacción, grado en que las necesidades del usuario están satisfechas cuando un producto o sistema se usa en un contexto de uso específico , Utilidad, grado en que un usuario está satisfecho con su percepción , logro de objetivos pragmáticos, incluidos los resultados de uso y las consecuencias del uso , Confianza, grado en que un usuario u otra parte interesada tiene confianza que un producto o sistema se comportará según lo previsto , Placer, grado en que un usuario obtiene placer de cumplir sus necesidades personales , Confort, grado en el que el usuario está satisfecho con el físico comodidad.

Libertad de riesgo, grado en que un producto o sistema mitiga el riesgo potencial para el estado económico, la vida humana, la salud o el medio ambiente

- Mitigación del riesgo económico, grado en que un producto o sistema mitiga el riesgo potencial para el estado financiero, operación eficiente, propiedad comercial, reputación u otros recursos en la intención contextos de uso
- Mitigación del riesgo de salud y seguridad, grado en que un producto o sistema mitiga el riesgo potencial para las personas en la intención contextos de uso
- Mitigación del riesgo ambiental, grado en que un producto o sistema mitiga el riesgo potencial para la propiedad o el medio ambiente en los contextos de uso previstos

SW_Quality.

Cobertura de contexto, grado en que un producto o sistema puede ser utilizado con efectividad, eficiencia, libertad de riesgo y satisfacción en ambos contextos de uso especificados y en contextos más allá de los inicialmente identificados explícitamente , Completitud del contexto, grado en que un producto o sistema puede ser utilizado con efectividad, eficiencia, libertad de riesgo y satisfacción en todos los contextos de uso especificados , Flexibilidad, grado en que un producto o sistema puede ser utilizado con efectividad, eficiencia, libertad de riesgo y satisfacción en contextos más allá de los especificados inicialmente en los requisitos .

Idoneidad funcional: grado en que un producto o sistema proporciona funciones que cumplen con los requisitos establecidos e implicaciones implícitas cuando se usan bajo condiciones específicas , (1) integridad funcional: grado en que el conjunto de funciones cubre todas las tareas especificadas y los objetivos del usuario , (2) corrección funcional: grado en que un producto o sistema proporciona los resultados correctos con el grado de precisión necesario , (3) idoneidad funcional: grado en que las funciones facilitan la realización de tareas y objetivos especificados. Ejemplo: A un usuario solo se le presenta los pasos necesarios para completar una tarea, excluyendo cualquier paso innecesario. Eficiencia de rendimiento: rendimiento relativo a la cantidad de recursos utilizados en virtud de lo establecido condiciones (los recursos pueden incluir otros productos de software, el software y el hardware) configuración del sistema y materiales (por ejemplo, papel de impresión, medios de almacenamiento). (a) comportamiento en el tiempo: grado en que los tiempos de respuesta y procesamiento y tasa de rendimiento de un producto o sistema, al realizar sus funciones, cumple con los requisitos , (b) utilización de recursos: grado en que las cantidades y tipos de recursos utilizados por un producto o el sistema, al realizar sus funciones, cumple con los requisitos (los recursos humanos son incluido como parte de la eficiencia) , (c) capacidad: grado en que los límites máximos de un producto o parámetro del sistema se encuentran requisitos (los parámetros pueden incluir la cantidad de elementos que se pueden almacenar, el número de usuarios concurrentes, el ancho de banda de comunicación, el rendimiento de las transacciones y el tamaño de base de datos), Idoneidad funcional:

Métricas de Completitud: integridad (a la especificación) de las funciones del Software: (1) # funciones disponibles / # funciones requeridas (proporción), (2) Corrección: corrección de las funciones, (3) #correcta resultados / # resultados (proporción), (4) Media y desviación estándar del error (al menos), (5) Adecuación: adecuación (a la especificación) de las funciones del software, (6) # funciones / funciones de # apropiadas (proporción).

Métricas de eficiencia de rendimiento: (1) Comportamiento de tiempo, (2) desviación media y estándar del tiempo necesario para completar una función (max, min también son útiles), (3) Funciones importantes, (4) todas las funciones (ponderadas), (5) Utilización de recursos: (6) media y desviación estándar de, (7) Uso de CPU, (8) Uso de memoria, (9) Número de archivo abierto.

Compatibilidad: grado en que un producto, sistema o componente puede intercambiar información con otros productos, sistemas o componentes, y / o realiza sus funciones requeridas, mientras comparte el mismo entorno de hardware o software. Coexistencia: grado en que un producto puede realizar sus funciones requeridas de manera eficiente mientras comparte un ambiente y recursos con otros productos, sin impacto perjudicial en ningún otro producto. Interoperabilidad: grado en que dos o más sistemas, productos o componentes pueden intercambiar información y uso la información que ha sido intercambiada Usabilidad: grado en que un producto o sistema puede ser utilizado por usuarios específicos para lograr objetivos específicos con eficacia, eficiencia y satisfacción en un contexto de uso específico (la usabilidad se puede especificar o medir como una característica de calidad del producto en términos de sus subcaracterísticas, o especificada o medida directamente por medidas que son un subconjunto de calidad en uso).

Reconocimiento de idoneidad: grado en que los usuarios pueden reconocer si un producto o sistema es apropiado para sus necesidades (la reconocibilidad de la idoneidad dependerá de la capacidad de reconocer la idoneidad de las funciones del producto o del sistema a partir de impresiones iniciales del producto o sistema y / o cualquier documentación). Capacidad de aprendizaje: grado en que un

producto o sistema puede ser utilizado por usuarios específicos para lograr objetivos específicos de aprender a usar el producto o sistema con efectividad, eficiencia, ausencia de riesgo y satisfacción en un entorno específico contexto de uso.

Operabilidad: grado en que un producto o sistema tiene atributos que facilitan su operación y control. Protección de error de usuario: grado en que un sistema protege a los usuarios contra errores de fabricación. Estética de la interfaz del usuario: grado en que una interfaz de usuario permite una interacción agradable y satisfactoria para el usuario. Accesibilidad: grado en que un producto o sistema puede ser utilizado por personas con la más amplia gama de características y capacidades para alcanzar un objetivo específico en un contexto de uso específico (El rango de capacidades incluye discapacidades asociadas con la edad).

Métricas de usabilidad: Reconocimiento de la adecuación. (1) media y desviación estándar del tiempo necesario para entender las funcionalidades del software , (2) Cuestionarios (escalas de Likert) , (3) Learnability: , (4) media y desviación estándar del tiempo necesario para aprender usar (hasta el 95% de las funcionalidades del software) ,(5) Cuestionarios (escalas de Likert) , (6) Operabilidad: fácil de operar , (7) # de funciones máximas de tres pasos / # de funciones (proporción) , (8) Estética de la interfaz de usuario: , (9) Cuestionarios (escalas de Likert).

Confiabilidad: grado en que un sistema, producto o componente realiza funciones especificadas bajo especificación condiciones para un período de tiempo específico (el desgaste no ocurre en el software. Las limitaciones en la confiabilidad son debidas a fallas en los requisitos, diseño e implementación, o debido a cambios contextuales). (1) madurez: grado en que un sistema, producto o componente satisface las necesidades de confiabilidad bajo condiciones normales operación. (2) disponibilidad: grado en que un sistema, producto o componente es operacional y accesible cuando requerido para el uso (Externamente, la disponibilidad se puede evaluar por la proporción de tiempo total durante el cual el sistema, producto o componente está en estado activo).(3) tolerancia a fallas: grado en que un sistema, producto o componente funciona según lo previsto a pesar de la presencia de fallas

de hardware o software. (4) capacidad de recuperación: grado en el que, en caso de interrupción o falla, un producto o sistema puede recuperar los datos directamente afectados y restablecer el estado deseado del sistema Seguridad: grado en que un producto o sistema protege la información y los datos para que las personas u otros los productos o sistemas tienen el grado de acceso a datos apropiado para sus tipos y niveles de autorización. (5) confidencialidad: grado en que un producto o sistema garantiza que los datos sean accesibles solo para aquellos autorizados para tener acceso. (6) integridad: grado en que un sistema, producto o componente impide el acceso no autorizado, o modificación de programas de computadora o datos. (7) no repudio: grado en que se puede demostrar que las acciones o eventos tuvieron lugar, de modo que los eventos o acciones no pueden ser repudiados más tarde. (8) responsabilidad: grado en que las acciones de una entidad se pueden rastrear de manera única a la entidad. (9) autenticidad: grado en que puede demostrarse que la identidad de un sujeto o recurso es la única reclamado.

Métricas de fiabilidad: Madurez. (1) MTBF (tiempo medio entre fallas). (2) Disponibilidad. (3) Corregir el tiempo de trabajo / tiempo de uso (proporción real: fracción). (4) Tolerancia a fallos. (5) # de funciones disponibles después del error e_i / # de función disponible (proporción). (6) Desviación media y estándar de las funciones disponibles después de un conjunto de errores. (7) Recuperabilidad. (8) media y desviación estándar del tiempo necesario para restaurar el sistema después de una interrupción. (9) media y desviación estándar de los datos perdidos debido a una interrupción (proporción).

Métricas de seguridad. Confidencialidad / Integridad: acceso no autorizado al software funciones. (1) Probabilidad de acceso no autorizado (definición operativa difícil). (2) # acceso no autorizado exitoso / # de acceso no autorizado (proporción). (3) Longitud de la clave de cifrado. Mantenibilidad: grado de efectividad y eficiencia con la cual un producto o sistema puede ser modificado por el objetivo mantenedores (las modificaciones pueden incluir correcciones, mejoras o adaptación del software a cambios en medio ambiente, y en requisitos y especificaciones funcionales). Modularidad: grado en que un sistema o programa

de computadora se compone de componentes discretos tales que un cambio a un componente tiene un impacto mínimo en otros componentes. Reutilización: grado en que un activo puede ser utilizado en más de un sistema o en la construcción de otros activos.

Analizabilidad: grado de efectividad y eficiencia con el que es posible evaluar el impacto en un producto o sistema de un cambio intencionado a una o más de sus partes, o para diagnosticar un producto por deficiencias o causas de fallas, o para identificar partes a ser modificadas. Modificabilidad: grado en que un producto o sistema puede ser modificado de manera efectiva y eficiente sin introducir defectos o degradación de la calidad del producto existente. Capacidad de comprobación: grado de eficacia y eficiencia con el que se pueden establecer criterios de prueba para un sistema, producto o componente y las pruebas se pueden realizar para determinar si esos criterios se han cumplido.

Portabilidad: grado de efectividad y eficiencia con la cual un sistema, producto o componente puede ser transferido desde un hardware, software u otro entorno operativo o de uso a otra portabilidad se puede interpretar como una capacidad inherente del producto o sistema para facilitar las actividades de portabilidad, o la calidad en el uso experimentada con el objetivo de portar el producto o sistema. Adaptabilidad: grado en que un producto o sistema puede ser adaptado efectiva y eficientemente para diferentes o en evolución hardware, software u otros entornos operacionales o de uso. Inestabilidad: grado de eficacia y eficiencia con el que un producto o sistema puede instalarse con éxito y / o desinstalado en un entorno específico (si el producto o sistema debe ser instalado por un usuario final, la capacidad de instalación puede afectar la idoneidad y operatividad funcional resultante). Capacidad de reemplazo: grado en que un producto puede reemplazar otro producto de software especificado para el mismo propósito en el mismo entorno (por ejemplo, la capacidad de reemplazo de una nueva versión de un producto de software es importante para el usuario al actualizar).

Métricas de mantenibilidad: (1) Modularidad. (2) # de llamadas entre clases

/ # de clases. (3) Analizabilidad. (4) # fuentes de errores descubiertas / mano de obra (proporción). (5) Complejidad ciclomática de Mc Cabe (indicador). (6) Modificabilidad. (7) # Errores corregidos fijos / mano de obra (proporción). (8) # nuevos errores descubiertos durante la prueba de regresión / # errores fijos – Testabilidad. (9) N° de cambios probados / mano de obra (proporción). Métricas de portabilidad: (1) Adaptabilidad. (2) N° de modificaciones después de haber cambiado el entorno operativo. (3) mano de obra. (4) Normalizado (KLOC / FP). (5) Instalabilidad. (6) media y desviación estándar del tiempo necesario para instalar el sw. (7) N° de problemas / cantidad de instalaciones (proporción). (8) Reemplazabilidad. (9) # de modificaciones. (10) mano de obra. (11) Tiempo de entrenamiento. (12) Normalizado (KLOC / FP). Hay 3 tipos de medidas 1. Directo, detectable sin la influencia de factores externos, como el entorno (hw y sw) en el que el software "enciende" el comportamiento y características de los usuarios, etc. Ejemplo de estas medidas son las que se encuentran en el código fuente (análisis estático y lectura de código) y los encontrados por una lectura de documentos específicos. 2. no directo, derivado de medidas de uno o más atributos.

Por ejemplo, las medidas relacionadas con los tiempos de respuesta dependen no solo del comportamiento del software en sí, sino también el entorno operativo en el que se ejecuta el software (hw y sw). 3. indicadores: algunas medidas pueden estimarse a partir de otras medidas (útiles en el caso de medidas que no pueden detectarse directamente). Por ejemplo, el tiempo de respuesta del software no se puede medir cuando el sw está todavía en un estado inalcanzable. Por lo tanto, la longitud del código se puede utilizar de una manera simple y áspera indicador de cuál será el tiempo de respuesta del producto en el entorno de utilizar.

ISO / IEC 25010: 2011 define: Un modelo de calidad en uso compuesto por cinco características (algunas de las cuales se subdividen en subcaracterísticas) que se relacionan con el resultado de la interacción cuando un producto se utiliza en un contexto particular de uso. Este modelo de sistema es aplicable al sistema humano-computadora completo, incluidos los sistemas informáticos en uso y los productos de software en uso. Un modelo de calidad del producto compuesto por ocho características (que se subdividen en subcaracterísticas) que se relacionan

con las propiedades estáticas del software y las propiedades dinámicas del sistema informático. El modelo es aplicable tanto a sistemas informáticos como a productos de software.

Las características definidas por ambos modelos son relevantes para todos los productos de software y sistemas informáticos. Las características y subcaracterísticas proporcionan terminología coherente para especificar, medir y evaluar la calidad del producto de sistema y software. También proporcionan un conjunto de características de calidad contra las cuales los requisitos de calidad establecidos se pueden comparar para que estén completos. Aunque el alcance del modelo de calidad del producto está destinado a ser software y sistemas informáticos, muchas de las características también son relevantes para sistemas y servicios más amplios. ISO / IEC 25012 contiene un modelo de calidad de datos que es complementario a este modelo. El alcance de los modelos excluye propiedades puramente funcionales, pero incluye la idoneidad funcional.

El alcance de la aplicación de los modelos de calidad incluye la especificación de soporte y evaluación de sistemas informáticos intensivos en software y software desde diferentes perspectivas por aquellos asociados con su adquisición, requisitos, desarrollo, uso, evaluación, soporte, mantenimiento, garantía y control de calidad, y auditoría. Los modelos pueden, por ejemplo, ser utilizados por desarrolladores, adquirentes, personal de control de calidad y evaluadores independientes, en particular los responsables de especificar y evaluar la calidad del producto de software. Las actividades durante el desarrollo del producto que pueden beneficiarse del uso de los modelos de calidad incluyen: identificar el software y los requisitos del sistema; validar la amplitud de una definición de requisitos; identificar el software y los objetivos de diseño del sistema; identificar los objetivos de prueba de software y sistema; identificar los criterios de control de calidad como parte del aseguramiento de la calidad; identificar los criterios de aceptación para un producto de software y / o un sistema informático intensivo en software; establecer medidas de características de calidad en apoyo de estas actividades.

Teoría del Estándar para métricas de calidad de software IEEE 1061-1998

El sponsor de este estándar fue el Comité de Normas de Ingeniería de Software del IEEE Computer Society. En febrero de 1984, se aprobó un proyecto para desarrollar un estándar para una metodología de métricas de calidad del software. y se formó un grupo de trabajo, porque no existía un estándar IEEE que cubriera el campo de métricas de calidad de software. En diciembre de 1992, el Consejo de Normas del IEEE aprobó IEEE Std 1061-1992. Fue publicado por IEEE el 12 de marzo de 1993. Este fue el primer estándar IEEE que se ocupó de las métricas de calidad. Es importante que los usuarios de este estándar entiendan que este es un estándar de proceso, y no un estándar que requiere métricas específicas para su uso.

La filosofía de este estándar es que una organización puede emplear cualquier grupo de métricas que considere más apropiadas para sus aplicaciones, siempre que se siga la metodología y las métricas estén validadas. Otra razón para este enfoque es que no hubo consenso sobre qué métricas para mandato de uso (las disposiciones de una norma son obligatorias, no opcionales). De acuerdo con este enfoque fue la carta constitutiva del Grupo de Trabajo, según lo dispuesto en la aprobación de la Junta de Estándares IEEE de la autorización del proyecto solicitud (PAR), que pedía que se desarrollara una metodología estándar.

Debido a la regla IEEE de que un estándar debe ser revisado o reafirmado dentro de los cinco años de la emisión, un PAR para una la revisión fue presentado y aprobado en 1998. La revisión fue reiniciada y recirculada, y los comentarios se resolvieron en 1998. La norma obtuvo la tasa de aprobación necesaria durante la votación y se presentó a la Junta de Normas IEEE-SA, que lo aprobó en diciembre de 1998.

La calidad del software es el grado en que el software posee una combinación deseada de atributos. Esta deseada combinación de atributos debe estar claramente definida; de lo contrario, la evaluación de la calidad se deja a la

intuición. Para el propósito de esta norma, definir la calidad del software para un sistema equivale a definir una lista de atributos de calidad de software requeridos para ese sistema. Para medir los atributos de calidad del software, un apropiado Conjunto de métricas de software debe ser identificado. El objetivo de las métricas de software es realizar evaluaciones a lo largo del ciclo de vida del software sobre si los requisitos de calidad del software se están cumpliendo. El uso de métricas de software reduce la subjetividad en la evaluación y control de la calidad del software al proporcionar una base cuantitativa para tomar decisiones sobre el software calidad.

Sin embargo, el uso de métricas de software no elimina la necesidad de juicio humano en las evaluaciones de software. Se espera que el uso de las métricas de software dentro de una organización o proyecto tenga un beneficio efecto haciendo que la calidad del software sea más visible. Más específicamente, el uso de la metodología de esta norma para medir la calidad permite a una organización (1) Lograr objetivos de calidad; (2) Establecer los requisitos de calidad para un sistema desde el principio; (3) Establecer criterios y estándares de aceptación; (4) Evaluar el nivel de calidad logrado contra los requisitos establecidos; (5) Detectar anomalías o señalar problemas potenciales en el sistema; (6) Predecir el nivel de calidad que se logrará en el futuro; (7) Monitorear los cambios en la calidad cuando se modifica el software; (8) Evaluar la facilidad de cambio en el sistema durante la evolución del producto; (9) Validar un conjunto de medidas. Para lograr estos objetivos, las métricas de procesos y productos deben estar representadas en el plan de métricas del sistema.

Este estándar se divide en cuatro cláusulas. La cláusula 1 proporciona el Alcance de esta norma. La cláusula 2 proporciona un conjunto de definiciones. La cláusula 3 proporciona una descripción general del marco para las métricas de calidad del software. La cláusula 4 proporciona una metodología para métricas de calidad de software. También en este estándar hay tres anexos que están incluidos solo con fines ilustrativos y de referencia. Esta norma proporciona una metodología para establecer los requisitos de calidad e identificar, implementar, analizar y validar las métricas de calidad de los procesos y productos de software.

Esta metodología se aplica a todo el software en todas las fases de cualquier ciclo de vida del software. Esta norma no prescribe métricas específicas.

Este estándar está destinado a aquellos asociados con la adquisición, desarrollo, uso, soporte, mantenimiento, y auditoría de software. El estándar está especialmente dirigido a aquellos que miden o evalúan la calidad del software. Este estándar puede ser utilizado por lo siguiente: (1) Un gerente de adquisición / proyecto para identificar, definir y priorizar los requisitos de calidad para un sistema; (2) Un desarrollador de sistemas para identificar rasgos específicos que deberían integrarse en el software para cumplir los requisitos de calidad; (3) Una organización de aseguramiento / control / auditoría de calidad y un desarrollador de sistemas para evaluar si la calidad se están cumpliendo los requisitos; (4) Un mantenedor del sistema para ayudar a implementar modificaciones durante la evolución del producto; (5) Un usuario para ayudar a especificar los requisitos de calidad para un sistema.

La aplicación de una metodología de métricas de calidad de software cumple con este estándar si se cumplen todas las disposiciones necesarias. Para los propósitos de esta norma, se aplican los siguientes términos y definiciones. IEEE Std 100-1996 e IEEE. Se debe hacer referencia a Std 610.12-1990 para términos no definidos en esta cláusula. Atributo: una propiedad física o abstracta medible de una entidad. Rango crítico: valores métricos utilizados para clasificar el software en las categorías de aceptable, marginal o inaceptable. Valor crítico: valor métrico de una métrica validada que se utiliza para identificar el software que tiene inaceptable calidad.

Métrica directa: una métrica que no depende de una medida de ningún otro atributo. Valor métrico directo: un objetivo numérico para un factor de calidad que debe cumplirse en el producto final. Por ejemplo, el tiempo medio hasta el fallo (MTTF) es una medida directa de la fiabilidad final del sistema. Medida: (A) Una forma de determinar o evaluar el valor comparándolo con una norma. (B) Para aplicar una métrica. Medición: el acto o proceso de asignar un número o categoría

a una entidad para describir un atributo de esa entidad Una figura, extensión o cantidad obtenida midiendo. Métrica: Ver: métrica de calidad del software.

El término métrica se usa en lugar del término métrica de calidad de software en este estándar. Marco de métricas: una ayuda de decisión utilizada para organizar, seleccionar, comunicar y evaluar los atributos de calidad requeridos para un sistema de software. Un desglose jerárquico de los factores de calidad, subfactores de calidad, y métricas para un sistema de software. Muestra de métricas: conjunto de valores de métrica que se extrae de la base de datos de métricas y se utiliza en la validación de métricas.

Validación métrica: el acto o proceso de asegurar que una métrica pronostica o evalúa confiablemente un factor de calidad. Valor métrico: una salida métrica o un elemento que está dentro del rango de una métrica. Métrica predictiva: métrica aplicada durante el desarrollo y utilizada para predecir los valores de un factor de calidad de software. Valor predictivo métrico: un objetivo numérico relacionado con un factor de calidad que debe cumplirse durante el desarrollo del sistema.

Este es un requisito intermedio que es un indicador temprano del rendimiento final del sistema. Por ejemplo, los errores de diseño o de código pueden ser pronosticadores tempranos de la confiabilidad final del sistema. Métrica de proceso: una medida utilizada para medir las características de los métodos, las técnicas y las herramientas empleado en el desarrollo, implementación y mantenimiento del sistema de software. Métrica del producto: una métrica utilizada para medir las características de cualquier producto intermedio o final del proceso de desarrollo de software. Atributo de calidad: una característica del software, o un término genérico que se aplica a factores de calidad, subfactoresde calidad valores métricos. Factor de calidad: un atributo de software orientado a la gestión que contribuye a su calidad. Muestra de factor de calidad: un conjunto de valores de factor de calidad que se extrae de la base de datos de métricas y se utiliza en la validación de métricas. Valor del factor de calidad: un valor de la métrica directa que representa un factor de calidad. Ver también: métrico valor.

Requisito de calidad: un requisito de que un atributo de software esté presente en el software para satisfacer un contrato, estándar, especificación u otro documento formalmente impuesto. Subfactor de calidad: descomposición de un factor de calidad o subfactor de calidad en sus componentes técnicos. Componente de software: término general utilizado para referirse a un sistema de software o un elemento, como un módulo, unidad, datos o documento. Métrica de calidad del software: una función cuyas entradas son datos de software y cuya salida es un solo valor numérico que puede interpretarse como el grado en que el software posee un atributo determinado que afecta su calidad. Esta definición difiere de la definición de métrica de calidad que se encuentra en IEEE Std 610.12-1990. Métrica validada: una métrica cuyos valores se han asociado estadísticamente con valores correspondientes al factor de calidad.

La calidad del software es el grado en que el software posee una combinación deseada de atributos de calidad. El propósito de las métricas del software es realizar evaluaciones a lo largo del ciclo de vida del software sobre si los requisitos de calidad del software se están cumpliendo. El uso de métricas de software reduce la subjetividad en la evaluación y el control de la calidad del software al proporcionar una base cuantitativa para tomar decisiones sobre la calidad del software.

Sin embargo, el uso de métricas de software no elimina la necesidad de juicio humano en las evaluaciones de software. Se espera que el uso de las métricas de software dentro de una organización o proyecto tenga un efecto beneficioso haciendo que la calidad del software sea más visible. Más específicamente, el uso de la metodología de esta norma para medir la calidad permite a una organización (1) Evaluar el logro de objetivos de calidad; (2) Establecer los requisitos de calidad para un sistema desde el principio; (3) Establecer criterios y estándares de aceptación; (4) Evaluar el nivel de calidad logrado contra los requisitos establecidos; (5) Detectar anomalías o señalar problemas potenciales en el sistema; (6) Predecir el nivel de calidad que se logrará en el futuro; (7) Monitorear los cambios en la calidad cuando se modifica el

software;(8) Evaluar la facilidad de cambio en el sistema durante la evolución del producto; (9) Validar un conjunto de medidas.

El marco de medición de la calidad del software que se muestra en la Figura 5 está diseñado para ser flexible. Permite adiciones, eliminaciones y modificaciones de factores de calidad, subfactores de calidad y métricas. Cada nivel puede ser expandido a varios subniveles. El marco se puede aplicar a todos los sistemas y se puede adaptar según corresponda sin cambiar el concepto básico.

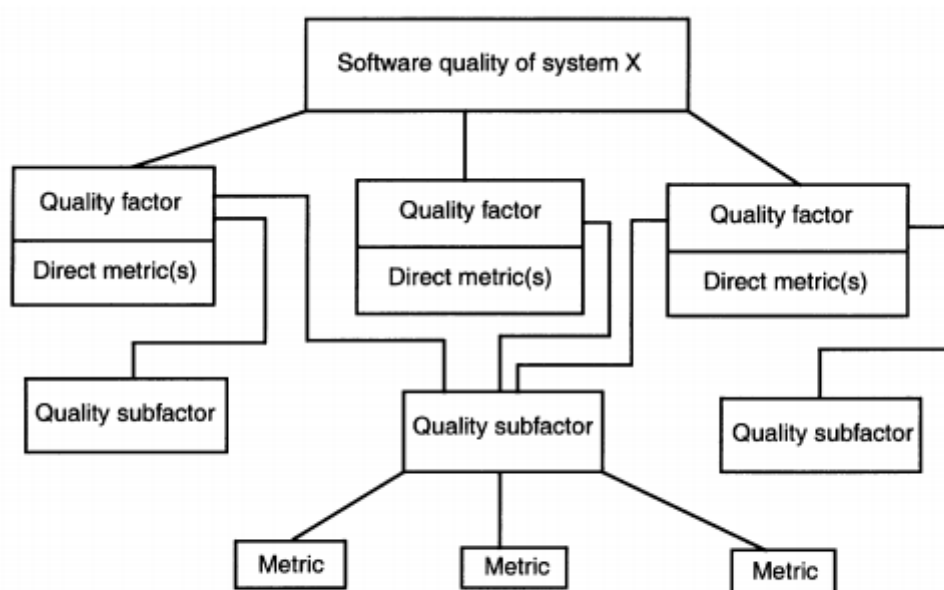


Figura 5. Marco de métricas de calidad de software. Obtenido de IEEE 1061-1998.

El primer nivel de la jerarquía del marco de medición de la calidad del software comienza con el establecimiento de los requisitos de calidad mediante la asignación de diversos atributos de calidad, que se utilizan para describir la calidad del sistema de entidad X. Todos los atributos que definen los requisitos de calidad son acordados por el equipo del proyecto, y entonces las definiciones están establecidas. Los factores de calidad que representan la administración y las vistas orientadas al usuario son luego asignados a los atributos.

Si es necesario, los subfactores de calidad se asignan a cada factor de calidad. Asociado con cada factor de calidad es una métrica directa que sirve como

una representación cuantitativa de un factor de calidad. Por ejemplo, una métrica directa para la confiabilidad del factor podría ser el tiempo medio hasta la falla (MTTF). Identificar una o más métricas directas y valores objetivos para asociar con cada factor, como un tiempo de ejecución de 1 hora, establecida por la administración del proyecto.

De lo contrario, no hay forma de determinar si el factor tiene logro. En el segundo nivel de la jerarquía están los subfactores de calidad que representan atributos orientados por software que indican calidad. Estos pueden obtenerse descomponiendo cada factor de calidad en atributos medibles de un software. Los subfactores de calidad son atributos independientes del software y, por lo tanto, pueden corresponder a más de un factor de calidad.

Los subfactores de calidad son atributos concretos del software que son más significativos que los factores de calidad para el personal técnico, como analistas, diseñadores, programadores, evaluadores y mantenedores. La descomposición de los factores de calidad en subfactores de calidad facilita la comunicación objetiva entre el gerente y el personal técnico con respecto a los objetivos de calidad.

En el tercer nivel de la jerarquía, los subfactores de calidad se descomponen en métricas utilizadas para medir el sistema, productos y procesos durante el ciclo de vida de desarrollo. Los valores métricos directos o valores de factor de calidad son típicamente no disponible o costoso de recolectar temprano en el ciclo de vida del software. De arriba a abajo, el marco facilita (1) Establecimiento de requisitos de calidad, en términos de factores de calidad, por parte de los gerentes al comienzo del sistema. Ciclo vital; (2) Comunicación de los factores de calidad establecidos, en términos de subfactores de calidad, a los personales; (3) Identificación de métricas relacionadas con los factores de calidad establecidos y los subfactores de calidad.

De abajo hacia arriba, el marco permite al personal gerencial y técnico obtener retroalimentación mediante. (4) Evaluar los productos y procesos de

software en el nivel de métricas; (5) Analizando los valores métricos para estimar y evaluar los factores de calidad.

Teoría de La metodología de métricas de calidad del software

Como explico Senevirathne, et al (2012) la metodología de métricas de calidad del software es un enfoque sistemático para establecer requisitos de calidad y identificar, implementar, analizar y validar las métricas de calidad del software del producto y del proceso para sistema de software. Comprende cinco pasos. Estos pasos están destinados a aplicarse de forma iterativa porque los conocimientos adquiridos al aplicar un paso pueden mostrar la necesidad de una evaluación adicional de los resultados de los pasos anteriores. Cada paso indica las actividades necesarias para lograr los resultados indicados.

Establecer los requisitos de calidad del software. El resultado de este paso es una lista de los requisitos de calidad. Las actividades para lograr este resultado son: (a) Identificar una lista de posibles requisitos de calidad Identifique los requisitos de calidad que son aplicables al sistema de software. Use experiencia organizativa normas, regulaciones o leyes requeridas para crear esta lista. Además, enumere otros requisitos del sistema que puede afectar la viabilidad de los requisitos de calidad.

Considere los requisitos contractuales y la adquisición preocupaciones, tales como restricciones de costos o cronogramas, garantías, requisitos de métricas del cliente y interés propio organizacional no descarte requisitos mutuamente excluyentes en este punto. Centrarse en la calidad factor / combinaciones de métricas directas en lugar de métricas predictivas. Asegúrese de que todas las partes involucradas en la creación y el uso del sistema participen en los requisitos de calidad proceso de identificación. (b) Determinar la lista de requisitos de calidad Califique cada uno de los requisitos de calidad enumerados por importancia. Determine la lista de los posibles requisitos de calidad haciendo lo siguiente: a) Encuestar a todas las partes involucradas. Discuta las prioridades relativas de los

requisitos con todos los involucrados.

Haga que cada grupo sopesa los requisitos de calidad con los demás requisitos del sistema y restricciones asegúrese de que todos los puntos de vista sean considerados. b) Cree la lista de requisitos de calidad. Resuelva los resultados de la encuesta en una sola lista de calidad requisitos. Los factores de calidad propuestos para esta lista pueden ser cooperativos o conflictivos relaciones. Resuelva cualquier conflicto entre los requisitos en este punto. Además, si la elección de los requisitos de calidad entra en conflicto con el costo, el cronograma o la funcionalidad del sistema, y alteran uno u otro. Tenga cuidado al elegir la lista deseada para asegurarse de que los requisitos sean técnicamente factibles. Razonable, complementario, alcanzable y verificable.

Obtener el acuerdo de todas las partes en esta finalista. (c) Cuantificar cada factor de calidad. Para cada factor de calidad, asigne una o más métricas directas para representar el factor de calidad y asigne directivos valores métricos para servir como requisitos cuantitativos para ese factor de calidad.). Use métricas directas para verificar el logro del requerimiento de calidad. La lista cuantificada de requisitos de calidad y sus definiciones nuevamente son aprobadas por todas las partes involucradas. (2) Identificar métricas de calidad del software. El resultado de este paso es un conjunto de medidas aprobado. Las actividades para lograr este resultado son: (a) Aplicar el marco de métricas de calidad del software.

La lista cuantificada de requisitos de calidad y sus definiciones nuevamente son aprobadas por todas las partes involucradas. Cree un cuadro de los requisitos de calidad basados en la estructura de árbol jerárquica que se encuentra en la Figura 1. En este punto, completa solo el nivel de factor de calidad. A continuación, descomponga cada factor de calidad en subfactores de calidad como indicado en la Cláusula 3. Continuar la descomposición en subfactores de calidad para tantos niveles como sea necesario hasta el nivel de subfactor de calidad está completo. Utilizando el marco de medidas de calidad del software, descomponga los subfactores de calidad en métricas.

Por cada métrica validada en el nivel métrico, asigne un valor objetivo y un valor crítico y rango que se debe alcanzar durante el desarrollo. Haga revisar el marco y los valores objetivos para las métricas revisados y aprobados por todas las involucradas fiestas. Utilice solo métricas validadas (es decir, métricas directas o métricas validadas con respecto a las métricas directas) para evaluar la calidad actual y futura de los productos y procesos (ver 4.5 para una descripción de la metodología de validación). Conservar métricas no validadas como candidatos para análisis futuros. Además, use solo métricas que sean asociados con los requisitos de calidad del proyecto de software.

Documente cada métrica utilizando el formato que se muestra en la Tabla 5. Realizar un análisis de costo-beneficio, Identificar los costos de implementación de las métricas Identifique y documente todos los costos asociados con cada métrica en el conjunto de medidas. Para cada métrica, estima y documenta los siguientes costos e impactos: a) Costos de utilización de métricas incurridos por la recopilación de datos; automatizar los cálculos de métricas; y aplicando, interpretar e informar los resultados; b) Costos de cambio de software incurridos al cambiar el proceso de desarrollo c) Cambios en la estructura organizativa de los costos incurridos al producir el software; d) Equipo especial requerido para implementar el plan de métricas; e) Capacitación requerida para implementar el plan de métricas.

Tabla 5.

Grupo de métricas

ItemDescription	
Name	Name given to the metric.
Costs	Costs of using the metric (see 4.2.2.1).
Benefits	Benefits of using the metric (see 4.2.2.2).
Impact	Indication of whether a metric can be used to alter or halt the project (ask, "Can the metric be used to indicate deficient software quality?").
Target value	Numerical value of the metric that is to be achieved in order to meet quality requirements. Include the critical value and the range of the metric.
Quality factors	Quality factors that are related to this metric.
Tools	Software or hardware tools that are used to gather and store data, compute the metric, and analyze the results.
Application	Description of how the metric is used and what its area of application is.
Data items	Input values that are necessary for computing the metric values.
Computation	Explanation of the steps involved in the metrics computation.
Interpretation	Interpretation of the results of the metrics computation (see 4.4.1). Considerations of the appropriateness of the metric (e.g., Can data be collected for this metric? Is the metric appropriate for this application?).

Training	required Training required to implement or use the metric.
Example	An example of applying the metric.
Validation	history Names of projects that have used the metric, and the validity criteria the metric has satisfied.
References	References, such as a list of projects and project details, giving further details on understanding or implementing the metric

Nota: Tomado de IEEE 1061-1992

Identifique y documente los beneficios que están asociados con cada métrica en el conjunto de métricas. Considerar incluir lo siguiente: a) Identificar objetivos de calidad y aumentar la conciencia de los objetivos. b) Proporcionar retroalimentación oportuna de problemas de calidad al proceso de desarrollo. c) Aumentar la satisfacción del cliente mediante la cuantificación de la calidad del software antes de que se entregue. d) Proporcionar una base cuantitativa para tomar decisiones sobre la calidad del software. e) Reducir los costos del ciclo de vida del software al mejorar la eficiencia del proceso.

Ajuste el conjunto de medidas. Pondere los beneficios, tangibles e intangibles, contra los costos de cada métrica. Si los costos exceden los beneficios de una métrica determinada, modifíquela o elimínela del conjunto de medidas. Por otro lado, para métricas que permanecen, haga planea cualquier cambio necesario en el proceso de desarrollo de software, estructura organizacional, herramientas y formación. En la mayoría de los casos, no será posible cuantificar los beneficios. En estos casos, ejercite el juicio al pesar beneficios cualitativos contra costos cuantitativos.

Ganar compromiso con el conjunto de métricas Haga que todas las partes involucradas revisen las métricas ajustadas. Haga que las métricas establecidas sean adoptadas y respaldadas por este grupo. (3) Implementar las métricas de calidad del software. Los resultados de este paso son una descripción de los elementos de datos, una matriz de trazabilidad y un plan de capacitación y programar. Las actividades para lograr este resultado son :(a) Definir los procedimientos de recopilación de datos Para cada métrica en el conjunto de medidas, determine los datos que se recopilarán y determine los supuestos eso se hará con los datos (p. ej., muestra aleatoria y medida subjetiva u objetiva).

Muestra el flujo de datos desde el punto de recolección hasta la evaluación de las métricas. Identificar herramientas y describir cómo se usarán. Describe los procedimientos de almacenamiento de datos. Establecer una matriz de trazabilidad entre las métricas y los elementos de datos. Identifique las entidades de la organización que participarán en la recopilación de datos, incluidos los responsables de monitorear la recolección de datos. Describa la capacitación y la experiencia requeridas para la recopilación de datos y la capacitación proceso para el personal involucrado.

Describa cada elemento de datos a fondo, utilizando el formato que se muestra en la figura 9. Prototipo del proceso de medición. Pruebe los procedimientos de recopilación de datos y cálculo de métricas en el software seleccionado que actuará como un prototipo. Seleccione muestras que sean similares a los proyectos en los que se usarán las métricas. Haga un análisis para determinar si los datos se recopilan de manera uniforme y si las instrucciones se han interpretado de manera coherente.

En particular, verifique los datos que requieren juicios subjetivos para determinar si las descripciones y las instrucciones son claras lo suficiente para garantizar resultados uniformes. Examine el costo del proceso de medición del prototipo para verificar o mejorar el análisis de costos. Use los resultados recopilados del prototipo para mejorar las descripciones métricas (consulte la 1) y las descripciones de elementos de datos (ver Tabla 9). (b) Recolectar los datos y calcular los valores métricos Utilizando los formatos de las Tablas 5y6, recopile y almacene datos en la base de datos de métricas del proyecto en el lugar apropiado. Tiempo en el ciclo de vida Verifique la exactitud y la unidad de medida correcta de los datos.

Tabla 6.

Descripción de ítems de datos.

Item	Description
Name	Name given to the data item.
Metrics	Metrics that are associated with the data item.
Definition	Straightforward description of the data item.
Source	Location of where the data item originates.
Collector	Entity responsible for collecting the data.
Timing	Time(s) in life cycle at which the data item is to be collected. (Some data items are collected more than once.)
Procedures	Methodology (e.g., automated or manual) used to collect the data.
Storage	Location of where the data are stored.
Representation	Manner in which the data are represented, e.g., precision and format (Boolean, Dimensionless, etc.).
Sample	Method used to select the data to be collected and the percentage of the available data that is to be collected.
Verification	Manner in which the collected data are to be checked for errors.
Alternatives	Methods that may be used to collect the data other than the preferred method. Integrity Person(s) or organization(s) authorized to alter the data item and under what Conditions.

Nota: Obtenido de IEEE 1061-1998.

Supervise la recopilación de datos. Si se utiliza una muestra de datos, verifique que los requisitos como la aleatoriedad, el mínimo tamaño de muestra y muestreo homogéneo. Verificar la uniformidad de los datos si más de una persona lo está coleccionando Calcule los valores de la métrica de los datos recopilados.

(4) Analice los resultados de las métricas del software. Los resultados de este paso son cambios en la organización y el proceso de desarrollo, que se indican a partir de interpretar y usar los datos de medición. Las actividades para lograr este resultado son: (a) Interpretar los resultados. Interpreta y registra los resultados. Analice las diferencias entre los datos métricos recopilados y los valores objetivos. Investigue las diferencias significativas. Identificar la calidad del software, Identifique y revise los valores de métrica de calidad para los componentes de software. Identificar valores de métricas de calidad que son fuera de los intervalos de tolerancia previstos (alta calidad baja o inesperada) para su posterior estudio.

Inaceptable la calidad puede manifestarse como una complejidad excesiva, documentación inadecuada, falta de rastreabilidad u otro atributo indeseable.

La existencia de tales condiciones es una indicación de que el software puede no satisfacer requisitos de calidad cuando se vuelva operativo. Debido a que muchas de las métricas directas que generalmente son no se pueden obtener interés durante el desarrollo del software (por ejemplo, métricas de confiabilidad), use métricas validadas cuando las métricas directas no están disponibles. Use métricas directas o validadas para los componentes y procesos de software. Compare los valores de métrica con los valores críticos de las métricas. Analizar en detalle los componentes del software cuyos valores se desvían de los valores críticos.

Dependiendo de los resultados del análisis, rediseño (aceptable la calidad se logra mediante el rediseño), chatarra (la calidad es tan pobre que el rediseño no es factible) o no cambia (las desviaciones de los valores métricos críticos se consideran insignificantes) los componentes del software. (b) Hacer predicciones de calidad del software. Use métricas validadas durante el desarrollo para hacer predicciones de valores de métricas directas. Comparar predicho valores de métricas directas con valores objetivo para determinar si marcar los componentes de software para obtener información detallada de análisis.

Haga predicciones para los componentes de software y los pasos del proceso. Analizar en detalle los componentes del software y los pasos del proceso cuyos valores de métrica directos pronosticados se desvían de los valores objetivo. (c) Asegurar el cumplimiento de los requisitos. Use métricas directas para garantizar el cumplimiento de los productos de software con los requisitos de calidad durante el sistema y test de aceptación. Use métricas directas para los componentes de software y los pasos del proceso. Compara estas métricas valores con valores objetivo de las métricas directas. Clasificar componentes de software y pasos de proceso cuyas mediciones se desvía de los valores objetivos como no cumplidores.

(5) Validar las métricas de calidad del software. El resultado de este paso es un conjunto de métricas validadas para realizar predicciones de valores de factores de calidad. Las actividades para lograr este resultado son: (a) Aplicar la metodología de validación. El objetivo de la validación de métricas es identificar las métricas de producto y proceso que pueden predecir especificadas los valores del factor de calidad, que son representaciones cuantitativas de los requisitos de calidad. Las métricas indicarán si los requisitos de calidad se han logrado o es probable que se logren en el futuro.

Cuando es posible medir los valores del factor de calidad en el punto deseado del ciclo de vida, las mediciones directas se usarán para evaluar la calidad del software. En algunos puntos del ciclo de vida, ciertos valores del factor de calidad (p. Ej., Fiabilidad) son no disponible. Se obtienen después del parto o al final del proyecto. En estos casos, las métricas validadas serán usadas al inicio de un proyecto para predecir los valores del factor de calidad.

La validación no significa una validación universal de las métricas para todas las aplicaciones. Más bien se refiere a la validación la relación entre un conjunto de métricas y un factor de calidad para una aplicación determinada. El historial de la aplicación de métricas indica que las métricas predictivas rara vez se validaron (es decir, era no demostrado a través del análisis estadístico que las métricas midieron las características del software que pretendió medir).

Sin embargo, es importante que las métricas predictivas se validen antes de que se utilicen para evaluar la calidad del software. De lo contrario, es posible que las métricas se apliquen incorrectamente (es decir, se pueden usar métricas que tienen poco o no tiene relación con las características de calidad deseadas). Aunque los subfactores de calidad son útiles al identificar y establecer factores de calidad y métricas, no se utilizan en la validación de métricas, porque el foco en la validación está en determinar si una estadística existe una relación significativa entre los valores métricos predictivos y los valores del factor de calidad. Los factores de calidad pueden verse afectados por múltiples variables.

Una métrica única, por lo tanto, puede no representar suficientemente

cualquier factor de calidad si ignora estas otras variables. (b) Aplicar criterios de validez. Para que se considere válida, una métrica predictiva debe tener un alto grado de asociación con los factores de calidad. Representa conformándose a los umbrales enumerados a continuación.

Una métrica puede ser válida con respecto a ciertos criterios de validez e inválidos con respecto a otros criterios. Con el fin de evaluar si una métrica es válida, los siguientes umbrales serán designados: V-cuadrado del coeficiente de correlación lineal, Coeficiente de correlación de rango B, Error de predicción, nivel de confianza, P-tasa de éxito. La descripción de cada criterio de validez se proporciona a continuación. Correlación. La variación en los valores del factor de calidad explicada por la variación en los valores métricos que está dado por el cuadrado del coeficiente de correlación lineal (R^2) entre la métrica y el correspondiente factor de calidad, deberá exceder V. Este criterio evalúa si existe una asociación lineal suficientemente fuerte entre un factor de calidad y una métrica para garantizar el uso de la medida como un sustituto del factor de calidad, cuando no es factible para usar el último. b) Seguimiento.

Si una métrica M está directamente relacionada con un factor de calidad F, para un producto o proceso dado, entonces un cambio en un valor de factor de calidad de FT1 a FT2, en los momentos T1 y T2, deberá ir acompañado de un cambio en el valor métrico de MT1 a MT2. Este cambio debe ser en la misma dirección (por ejemplo, si F aumenta, M aumenta). Si M está inversamente relacionado con F, entonces un cambio en F debe ir acompañado de un cambio en M en la dirección opuesta (por ejemplo, si F aumenta, M disminuye).

Para realizar esta prueba, calcule el coeficiente de correlación de rangos (r) a partir de n valores emparejados del factor de calidad y la métrica. Cada uno de los pares de factores de calidad / métricas deberá medirse en el mismo punto en el tiempo, y los n pares de los valores se miden en n puntos en el tiempo. El valor absoluto de r debe exceder B. Este criterio evalúa si una métrica es capaz de rastrear los cambios en la calidad del producto o proceso sobre el ciclo de vida.

Consistencia. Si los valores de factor de calidad F_1, F_2, F_n , correspondientes a productos o procesos 1, 2, n, tienen la relación $F_1 > F_2 > F_n$, entonces los valores métricos correspondientes tendrán la relación $M_1 > M_2 > M_n$. Para realizar esta prueba, calcule el coeficiente de correlación de rangos (r) entre valores emparejados (de los mismos componentes de software) del factor de calidad y la métrica.

El valor absoluto de r deberá exceder B . Este criterio evalúa si hay consistencia entre los rangos de los valores del factor de calidad de un conjunto de componentes de software y las clasificaciones de los valores de métrica para el mismo conjunto de componentes de software. Este criterio se utilizará para determinar si una métrica puede clasificar con precisión, por calidad, un conjunto de productos o procesos. d) Previsibilidad. Si se usa una métrica en el tiempo T_1 para predecir un factor de calidad para un producto o proceso determinado, deberá predecir un factor de calidad relacionado con una precisión de (1) donde es el valor real de F en el tiempo T_2 . Este criterio evalúa si una métrica es capaz de predecir un valor de factor de calidad con la precisión requerida. e) Poder discriminativo.

Una métrica debe ser capaz de discriminar entre componentes de software de alta calidad (por ejemplo, alto MTTF) y componentes de software de baja calidad (por ejemplo, bajo MTTF). El conjunto de métricas y los valores asociados con el primero deben ser significativamente más altos (o más bajos) que los asociados con este último. Este criterio evalúa si una métrica es capaz de separar un conjunto de componentes de software de alta calidad de un conjunto de componentes de baja calidad. Esta capacidad identifica valores críticos para métricas que se usará para identificar componentes de software que tienen una calidad inaceptable. Para realizar esta prueba, coloque el factor de calidad y los datos de la métrica en forma de una tabla de contingencia y calcule la estadística χ^2 .

Este valor excederá la estadística chi-cuadrada correspondiente a a.f) Confiabilidad. Una métrica debe demostrar la correlación, seguimiento, consistencia, predictibilidad y discriminación propiedades de potencia para al menos $P\%$ de las aplicaciones de la métrica. Este criterio se utiliza para garantizar

que una métrica haya pasado una prueba de validez en un número suficiente o porcentaje de aplicaciones para que haya confianza de que la métrica puede realizar su intención Funciona consistentemente. (2) Procedimiento de validación Identificar la muestra de los factores de calidad Se debe extraer una muestra de factores de calidad de la base de datos de métricas.

Identificar la muestra de métricas. Se tomará una muestra del mismo dominio (por ejemplo, los mismos componentes de software. La base de datos de métricas. Realizar un análisis estadístico El análisis descrito se realizará antes de que se use una métrica para evaluar la calidad de un producto o proceso, se validará con los criterios descritos Si una métrica no supera todas las pruebas de validez, solo se utilizará de acuerdo con los criterios prescrito por esas pruebas (por ejemplo, si solo supera la prueba de validez de seguimiento, se utilizará solo para el seguimiento la calidad de un producto o proceso).

Documentar los resultados. Los resultados documentados deben incluir la métrica directa, la métrica predictiva, los criterios de validación y los valores numéricos. Resultados, como mínimo. Revalidar las métricas. Una métrica validada puede no ser necesariamente válida en otros entornos o aplicaciones futuras. Por lo tanto, una métrica predictiva debe revalidarse antes de usarse para otro entorno o aplicación. Evaluar la estabilidad del medio ambiente La validación de métricas se realizará en un entorno de desarrollo estable (es decir, donde el lenguaje de diseño, el lenguaje de implementación o las herramientas de desarrollo del proyecto no cambian durante la vida del proyecto en el que la validación se realiza).

Una organización inicia la conformidad con los requisitos de validación de esta norma mediante la recopilación y validación de métricas en un proyecto (el proyecto de validación). Este proyecto será similar a aquel en que se aplican las métricas (el proyecto de la aplicación) con respecto a las habilidades de ingeniería de software, aplicación, tamaño y entorno de ingeniería de software. La validación y la aplicación de métricas se realizarán durante las mismas fases del ciclo de vida en diferentes proyectos. Por ejemplo, si la métrica X se recopila durante la fase de diseño del proyecto A y luego se valida con respecto al factor de calidad Y, que se

recoge durante la fase de operaciones del proyecto A, la métrica X sería utilizado durante la fase de diseño del proyecto B para predecir el factor de calidad Y con respecto a la fase de operaciones de proyecto B.

Importancia de la calidad en el desarrollo de software en RPG

La importancia de la calidad en sistemas de software se explica muy claramente: Muchos sistemas de software de hoy son muy complejos y tienen millones de líneas de código fuente, esos sistemas involucran miles de personas, su desarrollo dura mucho tiempo sobre diferentes ambientes de desarrollo, en tal ambiente se introducen muchos errores, suele haber pobre diseño y muchos problemas, todos estos factores hacen virtualmente imposible tener la total eliminación de problemas en el software y su impacto negativo, por ende, se llevan a cabo varias actividades de control de calidad de software para eliminar cierto tipo de problemas que conducen a dicho impacto negativo, o para reducir la probabilidad o gravedad del Impacto negativo cuando sea inevitable. Tian, J. (2007). En general las expectativas de calidad de la gente para los sistemas de software que usan son dobles: (1) los sistemas de software deben hacer lo que se supone deben hacer, en otras palabras, deben hacer las cosas correctas, (2) deben hacer las sus tareas específicas correcta o satisfactoriamente, en otras palabras, deben hacer bien las cosas.

Aspectos que comprende

Pressman (2010) Sugirió que la evaluación de la calidad del softwareenglobaunenfoque de gestión de calidad de ingeniería de softwareefectiva (métodos yHerramientas), revisiones técnicas formales que se apliqueduranteel proceso del software, una estrategia de prueba multiescalada, el control de ladocumentación del software y la gestión del cambio, un procedimiento queasegure un ajuste a los estándares de desarrollo de software y mecanismos demedición y generación de informes.

Pressman (2010, p 101) explico cómo el objetivo de la Ingeniería de software es producir un producto de alta calidad, y que un buen ingeniero debe utilizar

mediciones u herramientas modernas que evalúen la calidad del código fuente, cito varias medidas o métricas de calidad como corrección, facilidad de mantenimiento, integridad facilidad de uso, explico también que las métricas de calidad deben ayudar a la eliminación de defectos. La calidad implica el uso extensivo de marcos de la industria donde se establecen buenas prácticas de ingeniería para el análisis y control de la calidad de los productos de software, actualmente existen varios estándares entre los cuales está el ISO/IEC 9126-3, el cual provee métricas internas para medición de atributos de 6 características externas definidas por la ISO/IEC 9126-1, estas métricas permiten medir la calidad interna de un producto de software. Aplican a un producto de software no ejecutable, aplican durante las etapas de desarrollo, permiten medir la calidad de los entregables intermedios, permiten predecir la calidad del producto final, permiten al usuario iniciar acciones correctivas temprano en el ciclo de desarrollo. (Mendoza 2006).

El estándar IEEE Standard 1061, es una metodología para establecer requerimientos de calidad, identificación, implementación, análisis y validación de la calidad del producto y del proceso de software. (IEEE 2016). La calidad también implica la definición y uso de métricas de Software, que se definen como: La aplicación continúa de las técnicas basadas en la medición para el desarrollo de software. La calidad se puede entender como el proceso y sus productos a suministrar información significativa y actualizada sobre la gestión, junto con el uso de estas técnicas para mejorar el proceso y sus productos. (Goodman, (2013, p.32).

Para la selección de métricas de Software Hitz, Martin & Montazeri, Behzad. (1996) citaron el trabajo de Chidamber and Kemerer (1994), los cuales propusieron una suite de métricas de software preliminar adaptadas al diseño de software orientado a objetos en 1991, presentando varios atributos de productos internos. En un artículo más reciente, los autores presentan un tratado más completo sobre sus métricas candidatas, complementadas por el análisis de dos estudios empíricos y una breve guía sobre cómo aplicar sus métricas para apoyar el proceso de diseño orientado a objetos. En su propuesta de métricas sugieren una metodología para seleccionar características pertinentes de un producto de software para fungir como candidatas a ser métricas o KPIs para la evaluación de la calidad, aunque el estudio

se realizó para software orientado a objetos, sus principios se pueden adaptar a software diferentes.

1.4 Problema

1.4.1 Problema general

¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora de la calidad de Software hecho en RPG en una Institución Bancaria?

1.4.2 Problemas específicos

Problema específico 1

¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora de la performance del software hecho en RPG en una Institución Bancaria?

Problema específico 2

¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora de la mantenibilidad del software hecho en RPG en una Institución Bancaria?

Problema específico 3

¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora de las buenas prácticas en el software hecho en RPG en una Institución Bancaria?

Problema específico 4

¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora de las malas prácticas de codificación de Software en los programas hechos en RPG en una Institución Bancaria?

Problema específico 5

¿Cuál es el efecto de la implementación del Sistema QSOURCE en la seguridad del software hecho en RPG en una Institución Bancaria?

Problema específico 6

¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora de la complejidad del software hecho en RPG en una Institución Bancaria?

Problema específico 7

¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora de la obsolescencia del software hecho en RPG en una Institución Bancaria?

1.5 Justificación del estudio**1.5.1 Justificación teórica**

La presente investigación se justifica teóricamente puesto que se quiere evidenciar la necesidad del uso de herramientas especializadas para la ayuda en la toma de decisiones en áreas de tecnología en temas relacionados con calidad de las aplicaciones o sistemas que dan soporte a la operativa diaria del negocio y su generación de valor, en apoyo a los objetivos estratégicos empresariales.

Esta investigación se hace para promover el uso de una herramienta que permita a los gerentes de tecnología de información, visibilizar cuantitativamente la calidad de los sistemas de software desarrollados en la empresa o por terceros y así poder gerenciarlos y controlarlos de tal manera que los indicadores mostrados ayuden a la mejora de la productividad y eficiencia del área de TI.

1.5.2 Justificación Práctica

La presente investigación tiene una justificación práctica, puesto que la herramienta planteada podrá utilizarse para toma de decisiones por parte de la gerencia de desarrollo de software o de TI, de manera más acertada para el control de la calidad del software o sistemas que dan soporte a los procesos críticos de la cadena de valor de la organización donde se realizó la investigación.

Así mismo, es importante tener presente que en la institución en estudio no

se han realizado investigaciones que involucren determinar la asociación entre el uso de una herramienta de control de calidad y la mejora de la misma para software desarrollado en RPG.

1.5.3 Justificación metodológica

Respecto a la justificación metodológica, es importante destacar que los resultados de la presente investigación permitirán validar la aplicación del marco metodológico y de los instrumentos utilizados; los métodos, procedimientos y técnicas e instrumentos que han sido empleados en la investigación, una vez demostrada su validez y confiabilidad podrán ser utilizados en otros trabajos de investigación en donde se busque aportar al campo de las Tecnologías de Información y Comunicación en cuanto a calidad del software.

1.5.4 Justificación tecnológica

QSOURCE es una herramienta de software que llena el vacío en cuanto a herramientas CASE (ComputerAided Software Engineering) para sistemas de software desarrollados en lenguaje RPG de IBM para plataformas AS400 que se basan en características asociadas a tokens propios del lenguaje como métricas asociadas a calidad. QSOURCE mejorara la productividad en el desarrollo de software reduciendo costos y tiempo, permitirá a los ingenieros de software, desarrolladores y gerentes de desarrollo, visibilizar y cuantificar la calidad de sus sistemas desarrollados en RPG que constan de cientos de miles de líneas de código sobre plataformas IBM/AS400, y así poder tomar decisiones que les permitan alinear sus procesos a los objetivos del negocio, en cuanto a la generación de valor. Los resultados de esta investigación posibilitan el diseño y elaboración de técnicas, instrumentos y equipos para la producción de bienes económicos (software), que dinamizan los procesos productivos.

1.6 Hipótesis

1.6.1 Hipótesis general

El QSOURCE mejora la calidad de Software hecho en RPG en una Institución Bancaria.

1.6.2 Hipótesis específicas

Hipótesis específica 1

El QSOURCE mejora la performance del software hecho en RPG en una Institución Bancaria.

Hipótesis específica 2

El QSOURCE mejora la mantenibilidad del software hecho en RPG en una Institución Bancaria.

Hipótesis específica 3

El QSOURCE mejora el uso de buenas prácticas en el software hecho en RPG en una Institución Bancaria.

Hipótesis específica 4

El QSOURCE mejora la eliminación de malas prácticas en el software hecho en RPG en una Institución Bancaria.

Hipótesis específica 5

El QSOURCE mejora de la seguridad del software hecho en RPG de Software hecho en RPG en una Institución Bancaria.

Hipótesis específica 6

El QSOURCE mejora la simplificación de la complejidad del software hecho en RPG de Software hecho en RPG en una Institución Bancaria.

Hipótesis específica 7

El QSOURCE mejora la obsolescencia de códigos en el software hecho en RPG de Software hecho en RPG en una Institución Bancaria.

1.7 Objetivos

1.7.1 Objetivo general

Determinar el efecto de la implementación del Sistema QSOURCE en la mejora de la calidad de Software hecho en RPG en una Institución Bancaria.

1.7.2 Objetivos específicos

Objetivo específico 1

Determinar el efecto de la implementación del Sistema QSOURCE en la mejora del performance del software hecho en RPG en una Institución Bancaria

Objetivo específico 2

Determinar el efecto de la implementación del Sistema QSOURCE en la mejora de la mantenibilidad del software hecho en RPG en una Institución Bancaria.

Objetivo específico 3

Determinar el efecto de la implementación del Sistema QSOURCE en la mejora de buenas prácticas en el software hecho en RPG en una Institución Bancaria.

Objetivo específico 4

Determinar el efecto de la implementación del Sistema QSOURCE en la mejora de malas prácticas del software hecho en RPG en una Institución Bancaria.

Objetivo específico 5

Determinar el efecto de la implementación del Sistema QSOURCE en la mejora de la seguridad del software hecho en RPG en una Institución Bancaria.

Objetivo específico 6

Determinar el efecto de la implementación del Sistema QSOURCE en la

mejora de la complejidad del software hecho en RPG en una Institución Bancaria.

Objetivo específico 7

Determinar el efecto de la implementación del Sistema QSOURCE en la mejora de la obsolescencia del software hecho en RPG en una Institución Bancaria.

II Marco metodológico

2.1 Variables

Identificación de la variable

En la presente investigación se establecieron como variables de estudio la variable independiente a la herramienta QSOURCE y la variable dependiente cuantitativa discreta es la calidad del software.

Definición conceptual de la variable independiente QSOURCE

QSOURCE es una herramienta CASE, sobre el término se tiene que: Las herramientas CASE (ComputerAided Software Engineering), ayudan a los gestores y practicantes de ingeniería del software en todas las actividades asociadas a los procesos de software. Automatizan las actividades de gestión de proyectos, gestionan todos los productos de los trabajos elaborados a través del proceso, y ayuda a los ingenieros en el trabajo de análisis, diseño y codificación. Las herramientas CASE se pueden integrar dentro de un entorno sofisticado. Pressman (2010, P.597).

QSOURCE es una herramienta para la verificación estática de software, que está basada en la ausencia de ejecución de todo o parte del sistema. La ausencia de ejecución permite este tipo de análisis en cualquier etapa, la verificación estática puede ser manual o manejada con una herramienta.

Un analizador estático es una herramienta de software que analiza la estructura y texto de un programa sin ejecutarlo, permite verificar interfaces, construcciones incorrectas, variables no inicializadas, partes atípicas del programa, y puede generar documentación. Boulanger, J. (2013, p 9).

Definición conceptual de la variable dependiente Calidad del Software

Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente. (Pressman, 2010, p135).

Las dimensiones de la variable calidad del software son el performance, mantenibilidad, buenas prácticas, malas prácticas, seguridad, complejidad y obsolescencia.

Definición conceptual del indicador performance del Software

El performance se entiende como la característica del software que hace que su ejecución sea óptima en términos de tiempo y consumo de recursos.

El rendimiento se mide con base en la velocidad de procesamiento, el tiempo de respuesta, el uso de recursos, el conjunto y la eficiencia. Pressman (2010, p188).

Definición conceptual del indicador mantenibilidad del Software

La mantenibilidad, se define como “la facilidad con la que un sistema o componente de software puede ser modificado para corregir fallos, mejorar su funcionamiento u otros atributos o adaptarse a cambios del entorno. La mantenibilidad combina la capacidad del programa para ser ampliable (extensibilidad), adaptable y servicial (estos tres atributos se denotan con un término más común: mantenibilidad), y además que pueda probarse, ser compatible y configurable (capacidad de organizar y controlar los elementos de la configuración del software, y que cuente con la facilidad para instalarse en el sistema y para que se detecten los problemas. Pressman (2010, p188).

Definición conceptual del indicador buenas prácticas del Software

Son consideradas como buenas prácticas de ingeniería de software al compendio de acciones que buscan mantener lo más universal posible cualquier lenguaje de programación. Si bien es cierto que estos ya son universales y están establecidos, cada programador tiene su estilo y siguiéndole podría cometer el error de crear algo incomprensible para otros programadores. Pressman (2010, p690).

Definición conceptual del indicador malas prácticas del Software

Son consideradas como malas prácticas de programación al compendio de acciones que se salen del uso universal de cualquier lenguaje de programación. Son aquellas características que se relacionan con malas acciones según el consenso general de la comunidad de programación. Pressman (2010, p690).

Definición conceptual del indicador seguridad del Software

La seguridad del software es la característica que permite dilucidar si el código contiene instrucciones que van en detrimento de las normas de seguridad o son perjudiciales en el ambiente de ejecución del software. La seguridad del software es una actividad del aseguramiento del software que se centra en la

identificación y evaluación de los peligros potenciales que podrían afectarlo negativamente y que podrían ocasionar que falle todo el sistema. Si los peligros se identifican al principio del proceso del software, las características de su diseño se especifican de modo que los eliminen o controlen. Pressman (2010, p378).

Definición conceptual del indicador complejidad del Software

Según Pressman (2010, p417). Existe complejidad ciclomática y complejidad estructural del software, a su vez existen complejidad desde el punto de vista de instrucciones con características que hacen que el software sea difícil al tener muchos componentes y relaciones difíciles de seguir o rastrear.

Definición conceptual del indicador obsolescencia del Software

La obsolescencia del software es la característica que define cual instrucción o sentencia ya no debe usarse debido a nuevas actualizaciones de la misma que ya están a disposición. La historia de la ingeniería de software está salpicada de decenas de descripciones y metodologías de proceso, métodos de modelado y notaciones, herramientas y tecnología, todos ellos obsoletos. (2010, p94).

Definición operacional de la variable dependiente Calidad del Software

Para la medición de la variable calidad del software desarrollado en RPG se utilizará la herramienta QSOURCE, que analizará cada línea de los programas y analizará las características del lenguaje RPG asociadas a calidad del software que han sido usadas , las cuales se asocian a métricas para su evaluación según aparezcan o no en la ejecución del análisis, para ello se agrupan esas características en performance, complejidad, seguridad, obsolescencia, buenas prácticas, malas prácticas y mantenibilidad. La herramienta QSOURCE genera una base de datos con valores del análisis de cada métrica evaluada y genera reportes consolidados de los totales asignados como calificación.

2.2 Operacionalización de variables

Dimensiones	Indicadores	Ítems	Niveles y rangos
Complejidad	qc2le , bnmdir , disk , optimize , extpgm , import call , callb , extproc , callp , goto , exsr , if , dow , dou , for , qcmdexc		Bajo = (01-40) Medio = (41-51) Alto = (52-100)

	select * , /copy , alloc bitoff , biton , lineastabla , lineas hojas f lineas hojas i , lineas hojas e lineas hoja d , lineas hoja d lineas hoja c , lineas hoja p lineas hoja o , hoja free lineasvar.pointer total de import , runsqlstm , opnqryf lineas x rutina lineas x programa nivel anidamiento if1 nivel anidamiento dow nivel anidamiento dou nivel anidamiento for grtobjaut total procedimientos total, variables indep. total módulos ile total *srvgpm	
Seguridad	qcmd , strqm, strdfu, strpdm, chkobj, grtobjaut	Bajo = (01-40) Medio = (41-51) Alto = (52-100)
Obsolescencia	goto , add, cat, div, mult mvr , sqrt, z-add , z-sub , cab mhlzo, mhzoz, lineas hojas i, lineas hoja e, lineas hoja o	Bajo = (01-40) Medio = (41-51) Alto = (52-100)
Buenas Prácticas	qc2le, bnndir, actgrp, optimize, extpgm, import call, callb, extproc, callp goto, exsr, if ,dow, dou, for qcmdexc, select * , /copy %date, %time, %timesstamp %diff, %subdt, %day, %hours, %minutes, %months %months, %seconds, %years %char, %dec, chain, reade readpe, setll, setgt, %fields, programa, objetivo requerimiento, fecha memo fecha atencion, analista modificado por modificaciones maxlin fuentes *in lineas hoja f lineas hoja i lineas hoja e lineas hoja d lineas hojas c líneas hojas p lineas hoja o lineas hoja h lineas hoja free lineasvar.pointer Var. no usadas rutinas no usadas cod.quemado tot. export tot.import cpyf, runqry, crtpf, ovrdbf rclstg, runsqlstm, opnqryf strqm, chkobj, tot.linrutina tot. lin. por procedimito inz anidamiento if anidamiento dou	Bajo = (01-40) Medio = (41-51) Alto = (52-100)
Malas Práctica	qcmd, usrop, disk, printer	Bajo = (01-40)

Mantenibilidad	extpgm, close, add, cat div , goto import , call , callb , extproc , exsr , if , dow dou , for , qcmdexec , open , close , select * /copy , %date , %time , %timestamp , %diff %subdt , %day , %hours , %minutes , %month %mseconds , %second , %years , %char , %dec %fields , programa , objetivo , requerimiento , fecha memo fecha atención , analista , modificaciones , free , líneas hojas f , lineas hojas d , líneas hojas c , líneas hojas h , líneas hoja free , líneas def. var. poi , total variables no usadas total rutinas no usadas , total líneas código quemado total export , total import , total líneas rutina , Total,líneaprocedimiento, uso de inz () , nivel anida. if Nivel anida. do, nivel anida. Dou, nivel anida. dow Nivel anida. for , total procedimientos , total variables i total módulos ile , total módulos *srvpgm	Medio = (41-51) Alto = (52-100)
----------------	---	------------------------------------

2.3 Metodología

2.4 Tipo de estudio

La investigación obedece a un enfoque cuantitativo y el tipo de investigación es aplicado, respecto a este tipo de investigación se tiene que: la investigación aplicada tiene como objeto el estudio de un problema destinado a la acción. La investigación aplicada puede aportar hechos nuevos. Si proyectamos suficientemente bien nuestra investigación aplicada, de modo que podamos confiar en los hechos puestos al descubierto, la nueva información puede ser útil y estimable para la teoría.

La investigación aplicada, por su parte, concentra su atención en las posibilidades concretas de llevar a la práctica las teorías generales, y destinan sus esfuerzos a resolver las necesidades que se plantean la sociedad y los hombres. La resolución de problemas prácticos se circunscribe a lo inmediato, por lo cual su resultado no es aplicable a otras situaciones. La investigación aplicada puede integrar una teoría antes existente. La resolución de problemas echa mano típicamente de muchas ciencias, puesto que el problema es algo concreto y no se le puede resolver mediante la aplicación De principios abstractos de una sola

ciencia. (Baena, 2014, P.11).

2.5 Diseño

La presente investigación es un diseño experimental, del tipo o clase pre-experimental con un grupo experimental: El diseño más habitual de este tipo de investigación es el estudio antes-después (o pre-post) de un sólo grupo o con grupo de control no equivalente. Este tipo de diseño se basa en la medición y comparación de la variable respuesta antes y después de la exposición del sujeto a la intervención experimental. Los diseños antes-después con un sólo grupo permiten al investigador manipular la exposición, pero no incluyen un grupo de comparación. Cada sujeto actúa como su propio control. Espallargues Carreras Mireia, Almazán Sáez Cari, Pons Rafols JoanMV, erra Prat Mateu S. (2004)

Este diseño obedece al siguiente esquema:

G O1 X O2

Donde:

G = Grupo Experimental (conjunto de programas RPG)

O₁= Aplicación del Pre Test

O₂ = Aplicación del Post test

X = Experimento (QSOURCE)

A este diseño se le denomina de pre test-pos test con un solo grupo, a un grupo se le aplica una prueba previa al estímulo o tratamiento experimental, después se le administra el tratamiento y finalmente se le aplica una prueba posterior al estímulo. Este diseño ofrece una ventaja sobre otros: existe un punto de referencia inicial para ver qué nivel tenía el grupo en la(s) variable(s) dependiente(s) antes del estímulo. Es decir, hay un seguimiento del grupo.

2.6 Población, muestra y muestreo

2.6.1 Población

No hay población.

2.6.2 Muestra

No hay muestra.

2.6.3 Muestreo

La investigación considera el muestreo intencional; al respecto de este muestreo se tiene que todos los elementos muestrales de la población serán seleccionados bajo estricto juicio personal del investigador. En este tipo de muestreo el investigador tiene previo conocimiento de los elementos poblacionales. Aunque este muestreo es subjetivo, requiere que el investigador conozca los elementos muestrales, lo que permite que el muestreo sea significativo. Namakforoosh, (2005).

2.7 Técnicas e instrumentos de recolección de datos

2.7.1 Técnica

Hernández S (2010) enumero como otros métodos cuantitativos de recolección de datos, a los instrumentos mecánicos o electrónicos o instrumentos propios de cada disciplina, dando así validez al uso de la herramienta QSOURCE para la recolección y generación de datos de calidad.

2.7.2 Instrumentos

El reporte generado por la herramienta QSOURCE fue para la presente investigación la técnica e instrumento de recolección de información para la investigación, debido que técnicamente construido el reporte, registra con veracidad los valores generados por la herramienta de análisis para cada programa fuente analizado en sus diferentes métricas permitiendo incluso la validación de la hipótesis.

Ficha técnica del instrumento 1

Nombre del Instrumento: Reporte General de Estadísticas Análisis de calidad del código fuente

Autores: Giovanni Barrero Ortiz.

Año: 2017

Objetivo: Este reporte muestra los valores generados por la herramienta QSOURCE, para cada métrica evaluada en el programa RPG, muestra los datos referentes a la generación de los datos como:

- Librería
- Archivo fuente
- Programa
- Fecha
- Hora
- Métrica
- Valor generado
- Rango inicial
- Rango final

Población: 31 programas fuente RPG.

Número de ítem: $171 * 30 = 5130$

Tiempo de administración: 30 minutos

Normas de aplicación: La herramienta generara los datos según la parametrización hecha inicialmente.

Escala:

Niveles o rango: Se establecen los siguientes

<u>Nivel</u>	<u>Rango</u>	
Bajo	1	40
Medio	41	51
Alto	52	100

Los instrumentos propuestos se presentan en el Anexo 2.

Validez

Se refiere al grado que un instrumento de medición mide realmente la variable que pretende medir. La validez de los instrumentos está dada por el juicio de expertos y se corrobora con la validación de los instrumentos (Cuestionarios, reportes) que presenta resultados favorables en el juicio de expertos.

El tipo de validez de contenido ya que el instrumento refleja el dominio específico de la medición de la calidad.

Se utilizarán los siguientes aspectos de validación:

Indicadores	Criterios
Claridad	: Está formulado con lenguaje apropiado y específico.
Objetividad	: Está expresado en conductas observables.
Actualidad	: Adecuado al avance de la ciencia y la tecnología.
Suficiencia	: Comprende los aspectos en cantidad y calidad
Intencionalidad	: Adecuado para valorar aspectos de las estrategias
Consistencia	: Basado en aspectos teórico-científicos
Coherencia	: Entre los índices, indicadores y las dimensiones.
Metodología	: La estrategia responde al propósito del diagnóstico
Pertinencia	: El instrumento es funcional para el propósito de la
Investigación	

Tabla 7.

Relación de Validadores

Validador	Resultado
Dr. Jorge Rafael Díaz Dumont	Aplicable
Dra. Flores Castañeda Rosalynn	Aplicable
Dr. William Flores	Aplicable

Fiabilidad

Para el caso de la confiabilidad, la información reportada, y registrada en cada uno de los aplicativos, será la misma, cada vez que se acceda, no presentando ninguna variación; siendo la confiabilidad del 100%. Se usa la medida de estabilidad vía un pre test-post tes.

2.8 Métodos de análisis de datos

Para analizar cada una de las variables se ha utilizado del programa SPSS V. 22, y el lenguaje R, porcentajes en tablas y figuras para presentar la distribución de los datos, la estadística descriptiva, para la ubicación dentro de la escala de medición, para la contrastación de las hipótesis se aplica la estadística paramétrica.

Prueba de hipótesis

Para Torres (2007) "La hipótesis es un planteamiento que establece una relación entre dos o más variables para explicar y, si es posible, predecir probabilísticamente las propiedades y conexiones internas de los fenómenos o las causas y consecuencias de un determinado problema" p. (129)

Nivel de Significación:

Si es menor del valor 0.05, se dice que el coeficiente es significativo en el nivel de 0.05 (95% de confianza en que la correlación sea verdadera y 5% de probabilidad de error).

2.9 Aspectos éticos

Se seguirán los siguientes principios: reserva de identidad de los participantes. Citas de los textos y documentos consultados, no manipulación de resultado.

III Resultados

3.1 Análisis de resultados

Esta investigación tiene el propósito principal de evaluar la influencia del uso de la herramienta de software “Sistema QSOURCE en la mejora de la calidad de software en RPG” como herramienta CASE en el proceso de desarrollo de software dentro del banco AGROBANCO en lima Perú, se ejecutó la herramienta QSOURCE como instrumento de medición en dos ocasiones (pre-test y pos-test) a un grupo de programas que conforman la población de este estudio.

Para lograr el correcto análisis de los datos obtenidos con la aplicación del instrumento, se utilizaron las técnicas estadísticas descriptivas e inferenciales, mediante el uso del paquete estadístico SPSS para establecer la comparación del grupo antes y después de la ejecución del software QSOURCE.

De acuerdo al diseño utilizado (Pre-Experimental con pre-prueba y pos prueba y grupo intacto), el análisis se realizó en dos partes: el primer análisis consistió en conocer los resultados del pre-test para saber el nivel del grupo con respecto a la variable; seguidamente se hizo e tratamiento experimental a través del software QSOURCE; posteriormente se analizó el post-test y finalmente se compararon los resultados obtenidos en el pos-test y pre-test.

Es importante destacar que el nivel de influencia del uso del software QSOURCE como herramienta de control de calidad del software desarrollado en lenguaje RPG del Banco AGROBANCO, se determinó en función de los puntajes obtenidos con la aplicación de la prueba, valores que se ubican en los siguientes extremos: nota mínima = 0pts. Nota máxima 100 pts. En la siguiente tabla se muestran los programas y su tipo que se usaron como muestra para la investigación.

Tabla 8

Programas a analizar

Nombre tipo	descripción
AG0011	SQLRPGLE Mantenimiento a CadenasProduc
AG0011A	SQLRPGLE Mantenimiento a CadenasProduc
AG0012	SQLRPGLE Mantenimiento a Integrantes de
AG0121	SQLRPGLE Mantenimiento a Integrantes de
AG0122	RPGLE Mantenimiento a Integrantes de
AI0008	RPG Mantenimiento ParámetrosCont
AI0077	RPGLE Ajuste por inflación Cuentas I
BAP003	RPGLE Elimina registros con fecha p
BA0010	RPG Bills & Accounts Payable Vendo

BA0020	RPG	Bills Payable Master Payments
BA0030	RPG	Accts Payable Master Payments
BA0031	RPG	Accts Payable Master Payments
BA0040	RPG	Bills Payable Variable Payment
BA0050	RPG	Bills Payable Payments Authori
BA0055	RPG	Bill Payable Approve/Reject
BA0060	RPG	Bills Payable Payment Records
BA0070	RPG	Accts Payable Variable Payment
BA0080	RPG	Accts Payable Authorization/Re
BA0085	RPGLE	ReporteFacturas Fec. Pago (
BA0090	RPG	Accounts Payable Payment Recor
BA0100	RPG	Bill Payer Payments Selection
BA0105	RPG	Bills & A/P EODAY PmtSelectio
BA0110	RPG	Bills & Accounts Payable Check
BA0111	RPG	Bills & Accounts Payable Check
BA0112	RPG	accounts Payable Check Generat
BA0155	RPG	Accounts Payable Payments Sele
BA0160	RPG	Bills Payable Next Day Payment
BA0165	RPG	Bills Payable Payments Selecti
BA0180	RPG	Informes de Retencion a Provee
BA0500	RPG	Bills Payable Master Payment R
BA0510	RPG	Bills

Atendiendo a estas consideraciones y a la revisión del instrumento aplicado al inicio de este estudio (pre-test) se obtuvieron los siguientes resultados con el grupo participante (experimental) para cada una de las variables:

Tabla 9
Datos Pre Test

CALLIB	CALSRC	CALEMB	CALFEC	CALHOR	CALIF1	CALIF2	CALIF3	CALIF4	CALIF5	CALIF6	CALIF7
LIBGBO	FUENTES	AG0011	2017-04-07	10.06.20	50	27	41	27	100	41	60
LIBGBO	FUENTES	AG0011A	2017-04-07	10.06.20	50	28	43	27	100	41	60
LIBGBO	FUENTES	AG0012	2017-04-07	10.06.20	42	23	42	100	100	46	65
LIBGBO	FUENTES	AG0121	2017-04-07	10.06.20	44	33	52	100	100	46	65
LIBGBO	FUENTES	AG0122	2017-04-07	10.06.20	50	31	46	14	100	66	65
LIBGBO	FUENTES	AI0008	2017-04-07	10.06.20	53	33	50	25	100	58	68
LIBGBO	FUENTES	AI0077	2017-04-07	10.06.20	61	40	61	16	100	63	65

LIBGBO	FUENTES	BAP003	2017-04-07	10.06.20	33	45	31	50	100	65	100
LIBGBO	FUENTES	BA0010	2017-04-07	10.06.20	42	23	35	33	100	46	57
LIBGBO	FUENTES	BA0020	2017-04-07	10.06.20	45	23	38	22	100	46	83
LIBGBO	FUENTES	BA0030	2017-04-07	10.06.20	47	22	40	22	100	46	50
LIBGBO	FUENTES	BA0031	2017-04-07	10.06.20	50	22	42	22	100	46	50
LIBGBO	FUENTES	BA0040	2017-04-07	10.06.20	52	26	47	25	100	50	50
LIBGBO	FUENTES	BA0050	2017-04-07	10.06.20	44	23	37	28	100	46	54
LIBGBO	FUENTES	BA0055	2017-04-07	10.06.20	63	38	58	28	100	64	57
LIBGBO	FUENTES	BA0060	2017-04-07	10.06.20	52	29	47	33	100	50	57
LIBGBO	FUENTES	BA0070	2017-04-07	10.06.20	55	31	50	25	100	57	56
LIBGBO	FUENTES	BA0080	2017-04-07	10.06.20	50	23	44	33	100	42	57
LIBGBO	FUENTES	BA0085	2017-04-07	10.06.20	78	50	75	33	100	80	70
LIBGBO	FUENTES	BA0090	2017-04-07	10.06.20	52	29	47	33	100	50	57
LIBGBO	FUENTES	BA0100	2017-04-07	10.06.20	54	30	50	18	100	55	50
LIBGBO	FUENTES	BA0105	2017-04-07	10.06.20	50	18	45	12	100	55	56
LIBGBO	FUENTES	BA0110	2017-04-07	10.06.20	66	41	64	20	100	69	70
LIBGBO	FUENTES	BA0111	2017-04-07	10.06.20	57	33	53	20	100	63	70
LIBGBO	FUENTES	BA0112	2017-04-07	10.06.20	63	38	61	20	100	64	65
LIBGBO	FUENTES	BA0155	2017-04-07	10.06.20	55	25	47	22	100	53	90
LIBGBO	FUENTES	BA0160	2017-04-07	10.06.20	58	33	47	41	100	50	83
LIBGBO	FUENTES	BA0165	2017-04-07	10.06.20	50	21	43	22	100	46	83
LIBGBO	FUENTES	BA0180	2017-04-07	10.06.20	60	29	57	14	100	63	57
LIBGBO	FUENTES	BA0500	2017-04-07	10.06.20	60	40	50	55	100	53	59
LIBGBO	FUENTES	BA0510	2017-04-07	10.06.20	60	35	53	33	100	54	63

Donde Calif1 es performance, Calif2, Mantenibilidad, Calif3 es buenas prácticas, Calif4 es malas prácticas, Calif5 es seguridad, Calif6 es complejidad y Calif7 es obsolescencia. Atendiendo a estas consideraciones y a la revisión del instrumento aplicado en pos-test se obtuvieron los siguientes resultados con el grupo participante (experimental) para cada una de las variables:

Tabla 10

Datos Post Test

CALLIB	CALSRC	CALEMB	CALFEC	CALHOR	CALIF1	CALIF2	CALIF3	CALIF4	CALIF5	CALIF6	CALIF7
LIBGBO	FUENTES	AG0011	2017-04-17	18.31.05	54	50	60	33	100	50	70

LIBGBO	FUENTES	AG0011A	2017-04-17	18.31.05	54	50	60	33	100	50	70
LIBGBO	FUENTES	AG0012	2017-04-17	18.31.05	47	46	59	100	100	53	70
LIBGBO	FUENTES	AG0121	2017-04-17	18.31.05	50	48	61	100	100	53	70
LIBGBO	FUENTES	AG0122	2017-04-17	18.31.05	50	52	61	20	100	66	75
LIBGBO	FUENTES	AI0008	2017-04-17	18.31.05	62	65	71	50	100	69	80
LIBGBO	FUENTES	AI0077	2017-04-17	18.31.05	61	66	73	33	100	63	80
LIBGBO	FUENTES	BAP003	2017-04-17	18.31.05	45	87	44	56	100	77	100
LIBGBO	FUENTES	BA0010	2017-04-17	18.31.05	62	63	70	40	100	66	71
LIBGBO	FUENTES	BA0020	2017-04-17	18.31.05	55	54	63	33	100	57	63
LIBGBO	FUENTES	BA0030	2017-04-17	18.31.05	55	48	60	28	100	53	59
LIBGBO	FUENTES	BA0031	2017-04-17	18.31.05	57	48	61	28	100	53	59
LIBGBO	FUENTES	BA0040	2017-04-17	18.31.05	57	50	65	28	100	57	54
LIBGBO	FUENTES	BA0050	2017-04-17	18.31.05	52	52	61	33	100	58	63
LIBGBO	FUENTES	BA0055	2017-04-17	18.31.05	75	65	78	42	100	73	61
LIBGBO	FUENTES	BA0060	2017-04-17	18.31.05	66	60	71	50	100	61	71
LIBGBO	FUENTES	BA0070	2017-04-17	18.31.05	60	56	66	33	100	64	65
LIBGBO	FUENTES	BA0080	2017-04-17	18.31.05	66	60	72	40	100	61	76
LIBGBO	FUENTES	BA0085	2017-04-17	18.31.05	78	75	83	50	100	80	80
LIBGBO	FUENTES	BA0090	2017-04-17	18.31.05	66	60	71	50	100	61	71
LIBGBO	FUENTES	BA0100	2017-04-17	18.31.05	60	37	57	20	100	61	61
LIBGBO	FUENTES	BA0105	2017-04-17	18.31.05	66	57	76	16	100	70	72
LIBGBO	FUENTES	BA0110	2017-04-17	18.31.05	72	63	78	20	100	71	75
LIBGBO	FUENTES	BA0111	2017-04-17	18.31.05	70	63	77	20	100	71	75
LIBGBO	FUENTES	BA0112	2017-04-17	18.31.05	68	63	75	25	100	71	80
LIBGBO	FUENTES	BA0155	2017-04-17	18.31.05	63	52	65	33	100	64	68
LIBGBO	FUENTES	BA0160	2017-04-17	18.31.05	66	57	68	50	100	62	63
LIBGBO	FUENTES	BA0165	2017-04-17	18.31.05	61	54	68	33	100	61	68
LIBGBO	FUENTES	BA0180	2017-04-17	18.31.05	76	66	83	20	100	80	80
LIBGBO	FUENTES	BA0500	2017-04-17	18.31.05	75	71	76	71	100	69	80
LIBGBO	FUENTES	BA0510	2017-04-17	18.31.05	64	64	72	50	100	70	80

3.2 Resultados variable calidad

3.2.1 Análisis descriptivo variable Calidad Pretest Categorizado

Tabla 11

Estadísticos variable calidad pre test.

Estadísticos		
PRE TEST (CATEGORIZADO)		
N	Válido	31
	Perdidos	0

Esta tabla nos muestra el número de programas analizados, 31.

Tabla 12

Estadísticos variable calidad pre test.

PRE TEST (CATEGORIZADO)					
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	BAJA	6	19,4	19,4	19,4
	MEDIA	25	80,6	80,6	100,0
	Total	31	100,0	100,0	

En la tabla 12 se ve el número de programas analizados según la calificación y su porcentaje, 6 programas están con bajo performance y representa el 19.4%, 25 programas tienen calificación media en calidad y son el 80.6%, lo que dice que hay más programas con calidad media en el pre test.

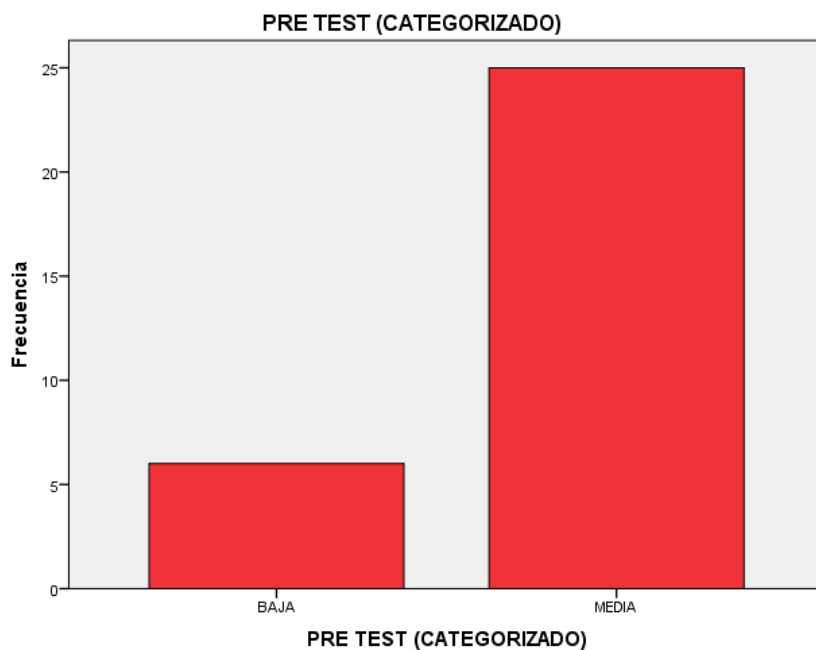


Figura6. Calidad pre test categorizado.

Tabla 13

Estadísticos variable calidad pre test.

Estadísticos		
PRE TEST		
N	Válido	31
	Perdidos	0

Estatabla nos muestra el número de programas analizados, 31.

Tabla 14

Tabla estadísticos calidad pre test.

	Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	47	3	9,7	9,7
	48	2	6,5	16,1
	49	1	3,2	19,4
	50	3	9,7	29,0
	51	2	6,5	35,5
	52	1	3,2	38,7
	53	4	12,9	51,6
	54	1	3,2	54,8
	55	1	3,2	58,1
	56	1	3,2	61,3
	57	2	6,5	67,7
	58	2	6,5	74,2
	59	2	6,5	80,6
	60	2	6,5	87,1
	61	2	6,5	93,5
	63	1	3,2	96,8
	69	1	3,2	100,0

Total	31	100,0	100,0
-------	----	-------	-------

Se muestra como frecuencia mayor el valor 53 en calidad con 4 programas que representan el 12.9%.

3.2.2 Estadísticos variable Calidad Pretest

Tabla 15

Estadísticos variable calidad pre test.

Estadísticos		
PRE TEST		
N	Válido	31
	Perdidos	0
Media		54,48
Mediana		53,00
Desviación estándar		5,452

Para la variable calidad pre test, se tienen los siguientes estadígrafos:

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **54,48 puntos**. Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica calidad en pre test 54.48.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **53.00**. El 50% de los programas de la muestra tienen una calificación mayor o igual a 53 en la variable calidad y 50% tienen menos o igual 53 puntos de calificación en su variable calidad en Pre test.

Medidas de Dispersión

La desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media. Corresponde a **5.452**.

3.2.3 Análisis descriptivo variable Calidad Post test Categorizado

Tabla 16

Tabla estadísticos calidad pos test categorizado.

Estadísticos		
<u>POS TEST (CATEGORIZADO)</u>		
N	Válido	31
	Perdidos	0
Media		2,23
Mediana		2,00
Desviación estándar		<u>,425</u>

Tabla 17

Estadísticos calidad pos test categorizado.

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	MEDIA	24	77,4	77,4	77,4
	ALTA	7	22,6	22,6	100,0
	Total	31	100,0	100,0	

En la tabla se ve el número de programas analizados según la calificación y su porcentaje, 24 programas está con calidad media y representa el 77.4%, 7 programas tienen calificación alta en calidad y son el 22.6%, lo que dice que hay más programas con calidad alta en el pos test que en el pre test. Lo que permite concluir que el uso de QSOURCE mejora la calidad de aplicaciones desarrolladas en RPG.

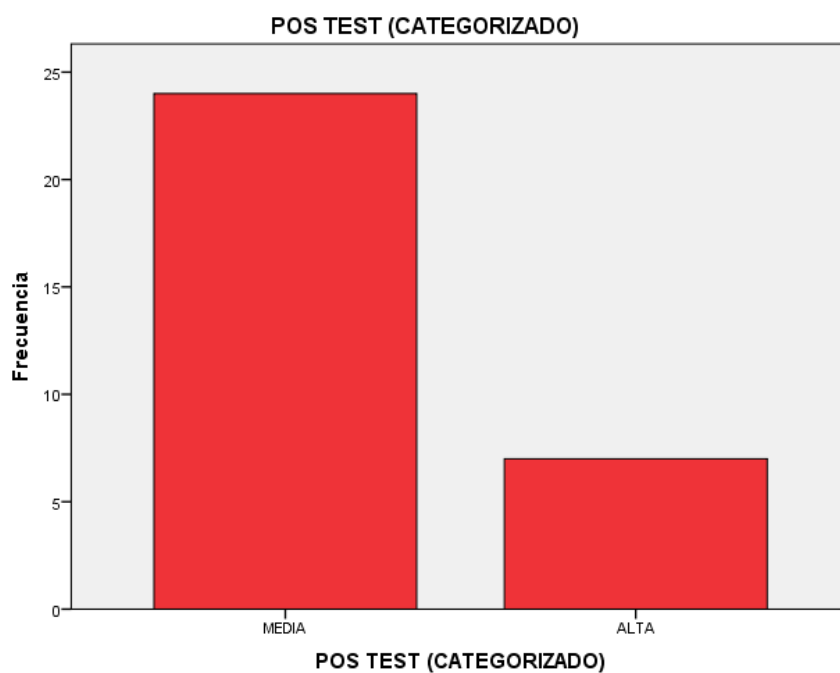


Figura7. Calidad pos test categorizado.

3.2.4 Estadísticos variable Calidad Post test

Tabla 18

Tabla estadísticos calidad pos test.

Estadísticos		
POS TEST		
N	Válido	31
	Perdidos	0
Media		66,16
Mediana		68,00
Desviación estándar		5,496

Para la variable calidad pos test, se tienen los siguientes estadígrafos:

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **66.16 puntos**. Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica calidad en pos test 66,16 contra 54.48 de pre test, lo que muestra una mejor calificación promedio de los programas en su variable calidad en el pos test.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **68.00**. El 50% de los programas de la muestra tienen una calificación mayor o igual a 68 en la variable calidad y 50% tienen menos o igual 68 puntos de calificación en su variable calidad en pos test contra 53 en el pre test, lo que permite concluir que en el pos test se ha incrementado el promedio de programas con mejor calificación de calidad.

Medidas de Dispersión

La desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media.

Corresponde a **5.496 en pos test** contra 5.452 del pre test, lo que permite concluir que los valores de la calificación de los programas de la muestra en su variable calidad tienen una mayor variabilidad en el pos test debido a los nuevos valores según las mejoras introducidas.

Tabla 19

Tabla frecuencia calidad pos test.

	Frecuencia	Porcentaje	Porcentaje	Porcentaje
Válido	a	e	válido	acumulado
57	1	3,2	3,2	3,2
58	2	6,5	6,5	9,7
59	1	3,2	3,2	12,9
60	3	9,7	9,7	22,6
61	2	6,5	6,5	29,0
63	1	3,2	3,2	32,3
64	2	6,5	6,5	38,7
65	1	3,2	3,2	41,9
67	2	6,5	6,5	48,4
68	7	22,6	22,6	71,0
69	2	6,5	6,5	77,4
71	3	9,7	9,7	87,1
72	1	3,2	3,2	90,3
73	1	3,2	3,2	93,5
77	1	3,2	3,2	96,8
78	1	3,2	3,2	100,0
Total	31	100,0	100,0	

Se muestra como frecuencia mayor el valor 68 en calidad con 7 programas que representan el 22.6%.

3.2.5 Prueba de Hipótesis variable Calidad Post test

Prueba T

Tabla 20

Estadísticas muestras emparejadas calidad pos test.

		Media	N	Desviación estándar	Media de error estándar
Par 1	PRE TEST	54,48	31	5,452	,979
	POS TEST	66,16	31	5,496	,987

La media para la variable calidad pos test se incremento en el pos test lo que sugiere una mejora en la calidad de los programas de la muestra modificados por la herramienta QSOURCE en el pos test.

PRE TEST - -11,677 3,700 ,665 -13,035 -10,320 -17,571 30 ,000
 POS TEST

Ho: la aplicación. QSOURCE no mejora significativamente la calidad
H1: la aplicación. QSOURCE mejora significativamente la calidad

Nivel significación: 0.05 (Max. Nivel error que quiero cometer en mi investigación, Si llevamos a cabo el mismo estudio 100 veces, aproximadamente 95 de los intervalos de confianza cubriría la verdadera media de la población)

Criterio: si $p\text{valor} < 0.05$ rechazo la Ho sino lo acepto

Según prueba de T. para muestra relacionada, el sig 0.000 < 0.05 entonces se rechaza Ho por lo tanto existe fuerte evidencia estadística para afirmar que existe diferencia en la calidad de las aplicaciones luego de la aplicación de QSOURCE concluyendo que mejora significativamente la calidad de 54.48 a 66.16.

3.2.6 Análisis de normalidad variable Calidad

Para que los test de pruebas emparejadas sean válidos la diferencia entre los valores debe ser distribuida normalmente.

Tabla 23

Prueba de normalidad calidad.

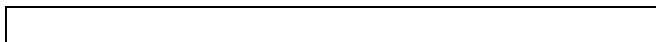
	POS TEST (CATEGORIZADO)	Kolmogorov-Smirnov ^a			Shapiro-Wilk		
		Estadístico	gl	Sig.	Estadístico	gl	Sig.
diff_calidad	MEDIA	,204	24	,011	,917	24	,051
	ALTA	,152	7	,200 [*]	,968	7	,880

*. Esto es un límite inferior de la significación verdadera.

a. Corrección de significación de Lilliefors

Prueba normalidad:

H0: existe normalidad en la distrib.
H1: no existenormalidad



Alfa = 0.05

Criterio: si pvalor <0.05 se rechaza la Ho sino se acepta

Se toma estadístico shapiro-wilk $n < 50$, pvalor $0.051 > 0.05$, para calificación media y pvalor $.880 > 0.05$, se acepta Ho, existe normalidad.

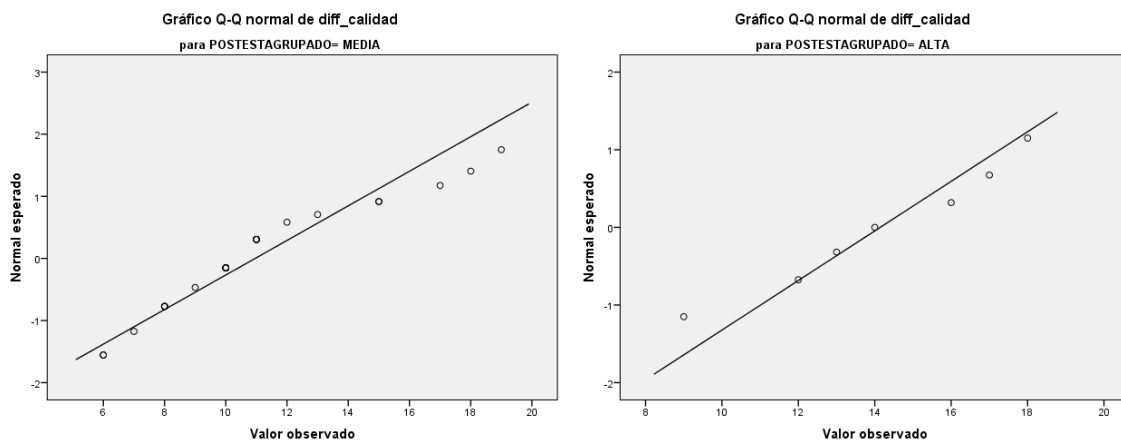


Figura9. Gráficos Q-Q normales calidad.

3.3 Resultados variable performance

3.3.1 Análisis descriptivo variable PerformancePretest Categorizado

Tabla 24

Estadísticos variablesperformance_pretest (categorizado)

PERFORMANCE_PRETEST (CATEGORIZADO)		
N	Válido	31
	Perdidos	0

Esta tabla nos muestra el número de programas analizados, 31.

Tabla 25

Performance Pretest (categorizado)

	Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	BAJO	1	3,2	3,2
	MEDIO	13	41,9	45,2
	ALTO	17	54,8	100,0
	Total	31	100,0	

En el grafico se ve el número de programas analizados según la calificación y su porcentaje, 1 programa está con bajo performance y representa el 3.2%, 13 programas tienen calificación medio en performance y son el 41.9% y hay 17 programas con calificación alto en performance y representan el 54%, lo que dice que hay más programas con un performance alto en el pre test.

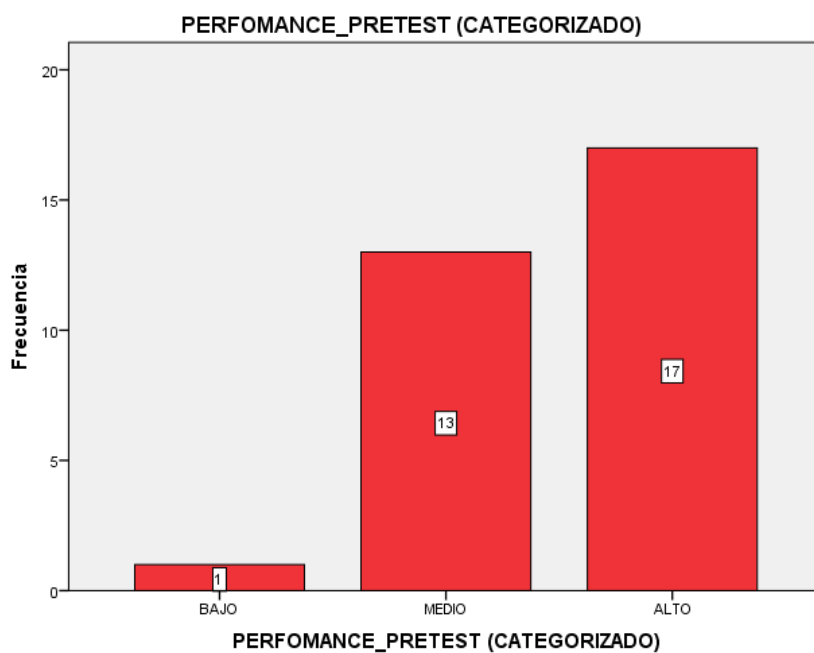


Figura10. Performance pre test categorizado.

Tabla 26

*Estadísticos performance Pretest***Estadísticos**

PERFORMANCE_PRETEST	
N	Válido 31

Perdidos	0
----------	---

Este grafico nos muestra el número de programas analizados, 31.

Tabla 27

Perfomance Pretest

PERFOMANCE_PRETEST					
		Frecuenci a	Porcentaj e	Porcentaje válido	Porcentaje acumulado
Válido	33	1	3,2	3,2	3,2
	42	2	6,5	6,5	9,7
	44	2	6,5	6,5	16,1
	45	1	3,2	3,2	19,4
	47	1	3,2	3,2	22,6
	50	7	22,6	22,6	45,2
	52	3	9,7	9,7	54,8
	53	1	3,2	3,2	58,1
	54	1	3,2	3,2	61,3
	55	2	6,5	6,5	67,7
	57	1	3,2	3,2	71,0
	58	1	3,2	3,2	74,2
	60	3	9,7	9,7	83,9
	61	1	3,2	3,2	87,1
	63	2	6,5	6,5	93,5
	66	1	3,2	3,2	96,8
	78	1	3,2	3,2	100,0
Total		31	100,0	100,0	

Se muestra como frecuencia mayor el valor 50 en PERFOMANCE con 7 programas que representan el 22.6%.

3.3.2 Estadísticos variable Performance Pretest

Tabla 28

Estadísticos performance Pretest

Estadísticos		
PERFORMANCE_PRETEST		
N	Válido	31
	Perdidos	0
Media		53,10
Mediana		52,00
Desviación estándar		8,611

Para la variable PERFORMANCE_PRETEST, se tienen los siguientes estadígrafos:

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **53,10 puntos**. Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica PERFORMANCE en pre test 53.10.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **52.00**. El 50% de los programas de la muestra tienen una calificación mayor o igual a 52 en la variable PERFORMANCE y 50% tienen menos o igual 52 puntos de calificación en su variable PERFORMANCE en Pre test.

Medidas de Dispersión

La Desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media. Corresponde a **8.611**).

3.3.3 Análisis descriptivo variable Performance Post test Categorizado

Tabla 29

Estadísticos performance pos test

Estadísticos

PERFORMANCE_POSTTEST (CATEGORIZADO)		
N	Válido	31
	Perdidos	0
Media		2,87
Mediana		3,00
Desviación estándar		,341

Tabla 30

Performance pos test (categorizado)

PERFORMANCE_POSTTEST (CATEGORIZADO)					
		Frecuenci a	Porcentaj e	Porcentaje válido	Porcentaje acumulado
Válido	MEDIO	4	12,9	12,9	12,9
	ALTO	27	87,1	87,1	100,0
	Total	31	100,0	100,0	

En la tabla se ve el número de programas analizados según la calificación y su porcentaje, 4 programas está con performance medio y representa el 12.9%,27

programas tienen calificación alto en performance y son el 87.1%, lo que dice que hay más programas con un performance alto en el pos test que en el pre test. Lo que permite concluir que el uso de QSOURCE mejora el performance de aplicaciones desarrolladas en RPG.

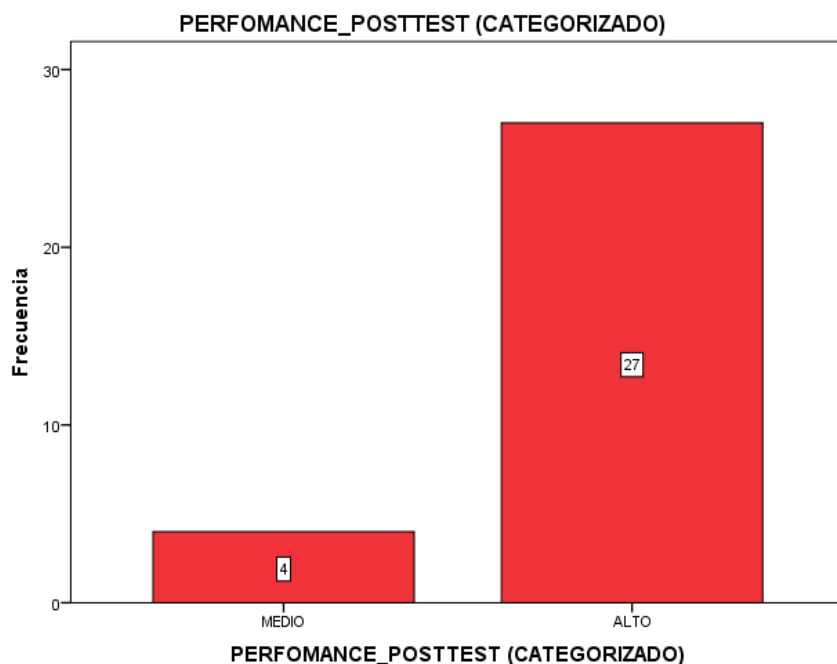


Figura 11. Performance pos test categorizado.

Tabla 31

Performance Pos Test

	Frecuencia	Porcentaje	Porcentaje	Porcentaje
	a	e	válido	acumulado
Válido	45	1	3,2	3,2
	47	1	3,2	6,5
	50	2	6,5	12,9
	52	1	3,2	16,1
	54	2	6,5	22,6
	55	2	6,5	29,0
	57	2	6,5	35,5
	60	2	6,5	41,9
	61	2	6,5	48,4
	62	2	6,5	54,8
	63	1	3,2	58,1
	64	1	3,2	61,3
	66	5	16,1	77,4
	68	1	3,2	80,6
	70	1	3,2	83,9
	72	1	3,2	87,1
	75	2	6,5	93,5
	76	1	3,2	96,8

78	1	3,2	3,2	100,0
Total	31	100,0	100,0	

Se muestra como frecuencia mayor el valor 66 en performance con 5 programas que representan el 16.1% en Pos Test contra 50 en PERFORMANCE con 7 programas que representan el 22.6% en pre test. Se muestra una diferencia incremental de programas con mejor calificación en la variable PERFORMANCE.

3.3.4 Estadísticos variable Performance Post test

Tabla 32

Estadísticos performance pos test

N	Válido	31
	Perdidos	0
Media		61,71
Mediana		62,00
Desviación estándar		8,680

Para la variable PERFORMANCE_POSTEST, se tienen los siguientes estadígrafos:

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **61.71 puntos**. Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica PERFORMANCE en pos test 61,71 contra 53.10 de pre test, lo que muestra una mejor calificación promedio de los programas en su variable PERFORMANCE en el pos test.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **62.00**. El 50% de los programas de la muestra tienen una calificación mayor o igual a 62 en la variable PERFORMANCE y 50% tienen menos o igual 62 puntos de calificación en su variable PERFORMANCE en pos test contra 52 en el pre test, lo que permite concluir que en el pos test se ha incrementado el promedio de programas con mejor calificación de PERFORMANCE.

Medidas de Dispersión

La desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media.

Corresponde a **8.680 en pos test** contra 8.611 del pre test, lo que permite concluir que los valores de la calificación de los programas de la muestra en su variable PERFORMANCE tienen una mayor variabilidad en el pos test debido a los nuevos valores según las mejoras introducidas.

3.3.5 Prueba de Hipótesis variable Performance Post test

Prueba T

Tabla 33

Estadísticas muestras emparejadas PERFORMANCE

		Media	N	Desviación estándar	Media de error estándar
Par 1	PERFORMANCE_PR ETEST	53,10	31	8,611	1,547
	PERFORMANCE_PO STTEST	61,71	31	8,680	1,559

La media para la variable PERFORMANCE_POSTEST se incremento en el pos test lo que sugiere una mejora en el PERFORMANCE de los programas de la muestra modificados por la herramienta QSOURCE en el pos test.

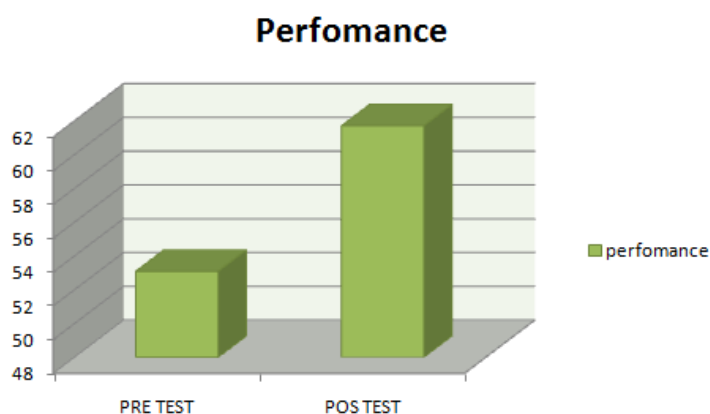


Figura12. Performance pre y pos test.

Tabla 34

Correlaciones muestras emparejadas PERFORMANCE

		N	Correlación	Sig.
Par 1	PERFOMANCE_PRETES T & PERFOMANCE_POSTTE ST	31	,820	,000

Tabla 35

Prueba de Muestras emparejadas PERFORMANCE

Diferencias emparejadas									
Par		Media	Desviación estándar	Media de error estándar	95% de intervalo de confianza de la diferencia		t	gl	Sig. (bilateral)
					Inferior	Superior			
1	PERFOMANCE _PRETEST - PERFOMANCE _POSTTEST	-8,613	5,181	,931	-10,513	-6,712	-9,255	30	,000

Ho: la aplicación. QSOURCE no mejora significativamente EL PERFORMANCE
H1: la aplicación. QSOURCE mejora significativamente EL PERFORMANCE

Nivel significación: 0.05 (Max. Nivel error que quiero cometer en mi investigación, Si llevamos a cabo el mismo estudio 100 veces, aproximadamente 95 de los intervalos de confianza cubriría la verdadera media de la población)

Criterio: si $p\text{valor} < 0.05$ rechazo la Ho sino lo acepto

Según prueba de T. para muestra relacionada, el sig 0.000 < 0.05 entonces se rechaza Ho por lo tanto existe fuerte evidencia estadística para afirmar que existe diferencia en el performance de la aplicación luego de la aplicación de QSOURCE

concluyendo que mejora significativamente EL PERFORMANCE.

3.3.6 Análisis de normalidad variable Performace

Para que los test de pruebas emparejadas sean válidos la diferencia entre los valores debe ser distribuida normalmente.

Tabla 36

Normalidad variable PERFORMANCE

Resumen de procesamiento de casos

	Casos					
	Válido		Perdidos		Total	
	N	Porcentaj e	N	Porcentaj e	N	Porcentaj e
DIFF_PERFOMAN CE	31	100,0%	0	0,0%	31	100,0%

Esta tabla muestra el cuadro de resumen que indica el número y porcentaje de casos analizados, son 31 y corresponde al 100%.

Tabla 37

Pruebas de Normalidad PERFORMANCE

Pruebas de normalidad

	Kolmogorov-Smirnov ^a			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
DIFF_PERFOMANCE	,128	31	,200*	,961	31	,305

*. Esto es un límite inferior de la significación verdadera.

a. Corrección de significación de Lilliefors

Prueba normalidad:

<p>H0: existe normalidad en la distrib. H1: no existenormalidad</p>
--

Alfa = 0.05

Criterio: si pvalor <0.05 se rechaza la Ho sino se acepta

Se toma estadístico shapiro-wilk $n < 50$, $p\text{-valor } 0.305 > 0.05$, se acepta H_0 , existe normalidad.

DIFF_PERFORMANCE

DIFF_PERFORMANCE Stem-and-Leaf Plot

Frequency Stem & Leaf

6.00	0 . 000444
13.00	0 . 5555666788889
7.00	1 . 0122344
4.00	1 . 5666
1.00	2 . 0

Stem width: 10.00

Each leaf: 1 case(s)

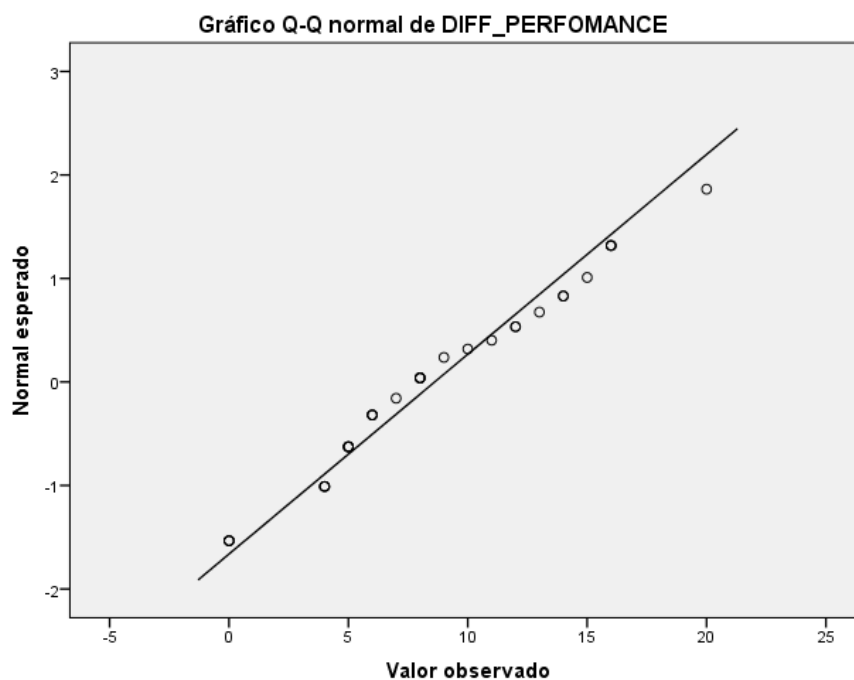


Figura13. Análisis normalidad performance.

La línea representa la distribución normal y los puntos la distribución de los datos de la muestra. Se concluye que los datos se comportan conforme a la normal ya que están ubicados sobre la línea (o lo más cercano posible).

3.4 Resultados Variable Buenas practicas

3.4.1 Análisis descriptivo variable Buenas Prácticas Pretest categorizado

Tabla 38

Estadísticos B. Practicas Pre Test (agrupado)

Estadísticos

B_PRACTICAS_PRETEST
(agrupado)

N	Válido	31
	Perdidos	0
Media		2,13
Mediana		2,00
Desviación estándar		,670

Tabla 39

B. Practicas Pre Test (agrupado)

		Frecuenci a	Porcentaj e	Porcentaje válido	Porcentaje acumulado
Válido	BAJO	5	16,1	16,1	16,1
	MEDIO	17	54,8	54,8	71,0
	ALTO	9	29,0	29,0	100,0
	Total	31	100,0	100,0	

En el grafico se ve el número de programas analizados según la calificación y su porcentaje, 5 programas está con bajo perfomance y representa el 16.5 %, 17 programas tienen calificación medio en perfomance y son el 54.8%, y 9 programas tienen calificación alta en B. Practicas con 29%, lo que dice que hay más programas con un perfomance medio en el pre test.

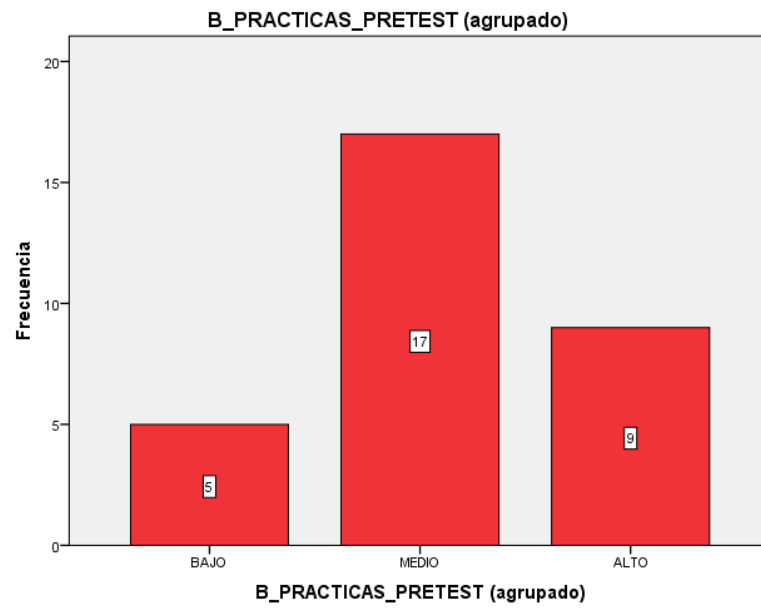


Figura14. Buenas prácticas pre test agrupado.

Tabla 40

Frecuencias B. Practicas Pre Test

	Frecuenci a	Porcentaj e	Porcentaje válido	Porcentaje acumulado
Válido 31	1	3,2	3,2	3,2
35	1	3,2	3,2	6,5
37	1	3,2	3,2	9,7
38	1	3,2	3,2	12,9
40	1	3,2	3,2	16,1
41	1	3,2	3,2	19,4
42	2	6,5	6,5	25,8
43	2	6,5	6,5	32,3
44	1	3,2	3,2	35,5
45	1	3,2	3,2	38,7
46	1	3,2	3,2	41,9
47	5	16,1	16,1	58,1
50	4	12,9	12,9	71,0
52	1	3,2	3,2	74,2
53	2	6,5	6,5	80,6
57	1	3,2	3,2	83,9
58	1	3,2	3,2	87,1
61	2	6,5	6,5	93,5
64	1	3,2	3,2	96,8
75	1	3,2	3,2	100,0
Total	31	100,0	100,0	

En la tabla se observa que la mayor frecuencia la tiene la calificación 47 de la variable Buenas prácticas en pre test con el 16.1 %.

3.4.2 Estadísticos variable Buenas Prácticas Pretest

Tabla 41

Estadísticos B. Practicas Pre Test

Estadísticos

B_PRACTICAS_PRETEST

N	Válido	31
	Perdidos	0
Media		48,26
Mediana		47,00
Desviación estándar		9,187

Para la variable B. PRACTICAS_PRETEST, se tienen los siguientes estadígrafos:

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **48.26 puntos**. Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica BUENAS PRACTICAS en pre test 48.26.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **47.00**. El 50% de los programas de la muestra tienen una calificación mayor o igual a 47 en la variable BUENAS PRACTICAS_PRETEST y 50% tienen menos o igual 47 puntos de calificación en su variable BUENAS PRACTICAS_PRETEST en Pre test.

Medidas de Dispersión

La Desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media). Corresponde a **9.187**.

3.4.3 Análisis descriptivo variable Buenas Prácticas post test categorizado

Tabla 42

Estadísticos B. Practicas Post Test (agrupado)

Estadísticos
B_PRACTICAS_POSTEST
(agrupado)

N	Válido	31
	Perdidos	0
Media		2,97
Mediana		3,00
Desviación estándar		,180

Tabla 43

B. Practicas Pos Test (agrupado)

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	MEDIO	1	3,2	3,2	3,2
	ALTO	30	96,8	96,8	100,0
	Total	31	100,0	100,0	

En la tabla se ve el número de programas analizados según la calificación y su porcentaje, 1 programas está con B. Practicas medio y representa el 3.2 %, 30 programas tienen calificación alto en B. Practicas y son el 96.8%, lo que dice que hay más programas con un performance alto en el pos test que en el pre test. Lo que permite concluir que el uso de QSOURCE mejora el uso de buenas prácticas en aplicaciones desarrolladas en RPG

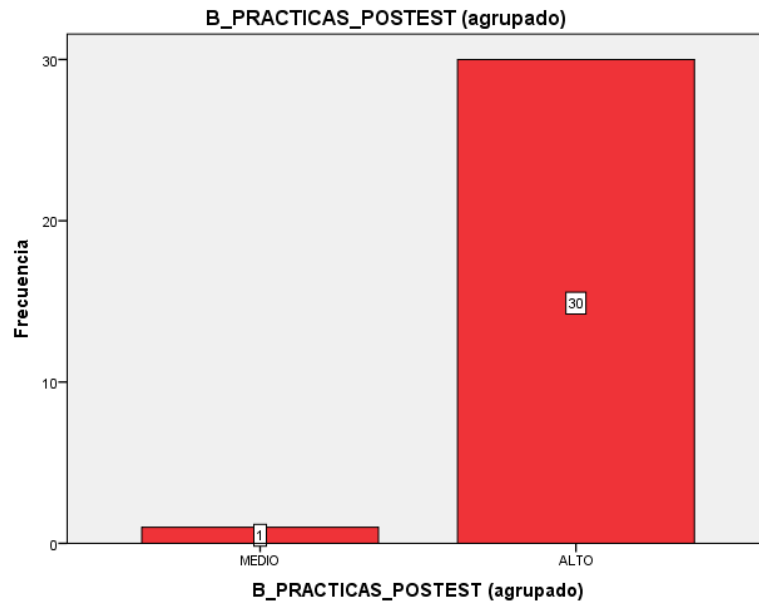


Figura15. Buenas prácticas pos test agrupado.

Tabla 44

Frecuencias B. Practicas Pos Test

	Frecuenci a	Porcentaj e	Porcentaje válido	Porcentaje acumulado
Válido 44	1	3,2	3,2	3,2
57	1	3,2	3,2	6,5
59	1	3,2	3,2	9,7
60	3	9,7	9,7	19,4
61	4	12,9	12,9	32,3
63	1	3,2	3,2	35,5
65	2	6,5	6,5	41,9
66	1	3,2	3,2	45,2
68	2	6,5	6,5	51,6
70	1	3,2	3,2	54,8
71	3	9,7	9,7	64,5
72	2	6,5	6,5	71,0
73	1	3,2	3,2	74,2
75	1	3,2	3,2	77,4
76	2	6,5	6,5	83,9
77	1	3,2	3,2	87,1
78	2	6,5	6,5	93,5
83	2	6,5	6,5	100,0
Total	31	100,0	100,0	

En la tabla se observa que la mayor frecuencia la tiene la calificación 61 con el 12.9% en pos test contra 47 de la variable Buenas prácticas en pre test con el 16.1 %. Lo que permite concluir que la frecuencia en el pos test refleja una mejora de la calificación promedio de los programas mejorados con la herramienta QSOURCE.

3.4.4 Estadísticos variable Buenas Prácticas pos test

Tabla 45

Estadísticos B. Practicas Pos Test

Estadísticos

B_PRACTICAS_POSTEST

N	Válido	31
	Perdidos	0
Media		67,90
Mediana		68,00
Desviación estándar		8,580

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **67.90 puntos.** Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica BUENAS PRACTICAS en pos test 67.90 contra 48.26 de pre test, lo que muestra una mejor calificación promedio de los programas en su variable BUENAS PRACTICAS en el pos test.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **68.00.** El 50% de los programas de la muestra tienen una calificación mayor o igual a 68 en la variable BUENAS PRACTICAS y 50% tienen menos o igual 68 puntos de calificación en su variable BUENAS PRACTICAS en pos test contra 47 en el pre test, lo que permite concluir que en el pos test se ha incrementado el promedio de programas con mejor calificación de BUENAS PRACTICAS.

Medidas de Dispersión

La Desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media. Corresponde a **8.580 en pos test** contra 9.187 del pre test, lo que permite concluir que los valores de la calificación de los programas de la muestra en su variable BUENAS PRACTICAS tienen una mayor variabilidad en el pos test debido a los nuevos valores según las mejoras introducidas

3.4.5 Prueba hipótesis variable Buenas Prácticas post test Prueba T

Tabla 46

Estadísticas de Muestras emparejadas B. Practicas

		Media	N	Desviación estándar	Media de error estándar
Par 1	B_PRACTICAS_PR E TEST	48,26	31	9,187	1,650
	B_PRACTICAS_PO S TEST	67,90	31	8,580	1,541

La media para la variable B_PRACTICAS_POSTEST se incrementó en el pos test lo que sugiere una mejora en B_PRACTICAS de los programas de la muestra modificados por la herramienta QSOURCE en el pos test.

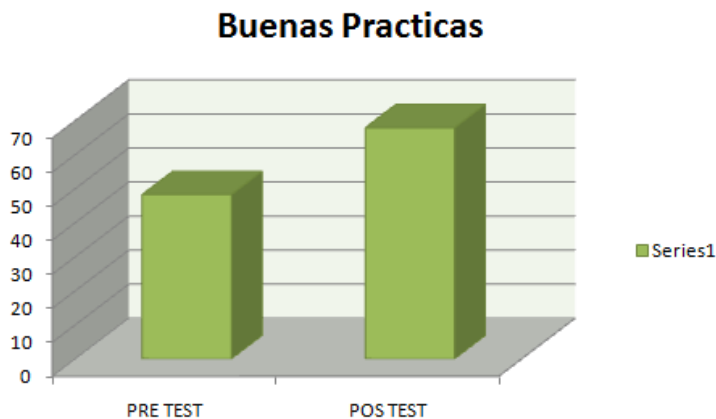


Figura16. Buenas prácticas pre y pos test.

Tabla 47

Correlación de muestras emparejadas B. Practicas

		N	Correlación	Sig.
Par 1	B_PRACTICAS_PRETEST & B_PRACTICAS_POSTEST	31	,731	,000

Tabla 48

Prueba de muestras emparejadas B. Practicas

Diferencias emparejadas									
		Media	Desviación estándar	Media de error estándar	95% de intervalo de confianza de la diferencia		t	gl	Sig. (bilateral)
Par 1	B_PRACTICA S_PRETEST - B_PRACTICA S_POSTEST	-19,645	6,540	1,175	-22,044	-17,246	-16,725	30	,000

<p>Ho: la aplicación. QSOURCE no mejora significativamente LAS BUENAS PRACTICAS H1: la aplicación. QSOURCE mejora significativamente LAS BUENAS PRACTICAS</p>
--

Nivel significación: 0.05 (Max. Nivel error que quiero cometer en mi investigación, Si llevamos a cabo el mismo estudio 100 veces, aproximadamente 95 de los intervalos de confianza cubriría la verdadera media de la población)

Criterio: si $p\text{valor} < 0.05$ rechazo la H_0 sino lo acepto

Según prueba de T. para muestra relacionada, el sig 0.000 < 0.05 entonces se rechaza H_0 por lo tanto existe fuerte evidencia estadística para afirmar que existe diferencia en las buenas prácticas luego de la aplicación de QSOURCE, concluyendo que mejora significativamente LAS BUENAS PRACTICAS de programas desarrollados en RPG.

3.4.6 Análisis de Normalidad

Para que los test de pruebas emparejadas sean válidos la diferencia entre los valores debe ser distribuida normalmente.

Tabla 49

Resumen de procesamiento de casos B. Practicas

	Casos					
	Válido		Perdidos		Total	
	N	Porcentaje	N	Porcentaje	N	Porcentaje
DIFF_B_PRACTICAS	31	100,0%	0	0,0%	31	100,0%

Esta tabla muestra el cuadro de resumen que indica el número y porcentaje de casos analizados, son 31 y corresponde al 100%.

Tabla 50

Pruebas de normalidad B. Practicas

	Kolmogorov-Smirnov ^a			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
DIFF_B_PRACTICAS	,102	31	,200*	,985	31	,936

*. Esto es un límite inferior de la significación verdadera.

a. Corrección de significación de Lilliefors

Prueba normalidad:

Ho: existe normalidad en la distrib.
H1: no existenormalidad

Alfa = 0.05

Criterio: si pvalor <0.05 se rechaza la Ho sino se acepta

Se toma estadístico shapiro-wilk n < 50, pvalor 0.936 > 0.05, se acepta Ho, existe normalidad.

DIFF_B_PRACTICAS

DIFF_B_PRACTICAS Stem-and-Leaf Plot

Frequency Stem & Leaf

3.00	0 . 789
4.00	1 . 2344
9.00	1 . 567788999
8.00	2 . 00114444
5.00	2 . 55668
1.00	3 . 1
1.00	3 . 5

Stem width: 10.00

Each leaf: 1 case(s)

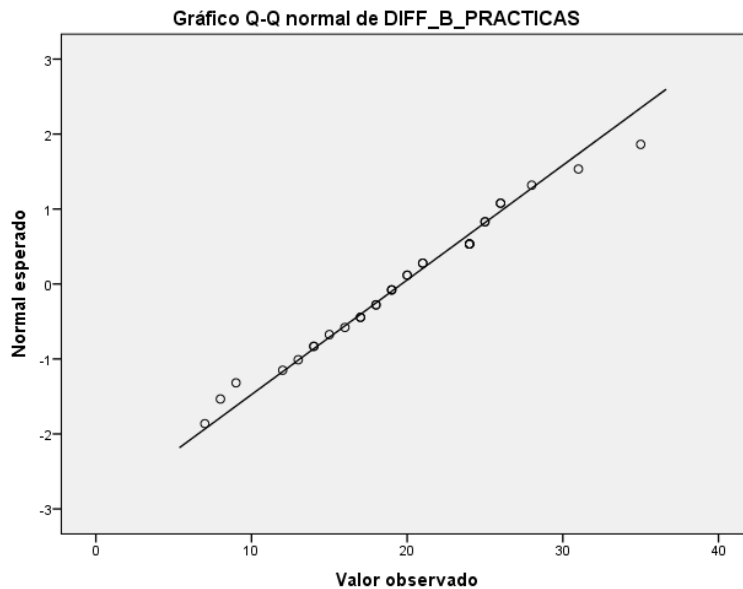


Figura17. Normalidad Buenas Prácticas.

La línea representa la distribución normal y los puntos la distribución de los datos de la muestra. Se concluye que los datos se comportan conforme a la normal ya que están ubicados sobre la línea (o lo más cercano posible).

3.5 Resultados variable Malas Prácticas

3.5.1 Análisis descriptivo variable Malas Prácticas Pretest categorizado

Tabla 51

Estadísticos Malas Prácticas (agrupado)

M_PRATICAS_PRETEST		
(agrupado)		
N	Válido	31
	Perdidos	0
Media		1,13
Mediana		1,00
Desviación estándar		,428

Tabla 52

Malas prácticas pre test (agrupado)

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	BAJO	28	90,3	90,3	90,3
	MEDIO	2	6,5	6,5	96,8
	ALTO	1	3,2	3,2	100,0
	Total	31	100,0	100,0	

En la tabla se ve el número de programas analizados según la calificación y su porcentaje, 28 programas está con malas prácticas y representa el 90.3 %, 2 programas tienen calificación medio en malas prácticas y son el 6.5%, y 1 programas tienen calificación alta en B. Practicas con el 3.2, lo que dice que hay más programas con malas prácticas bajo en el pre test.

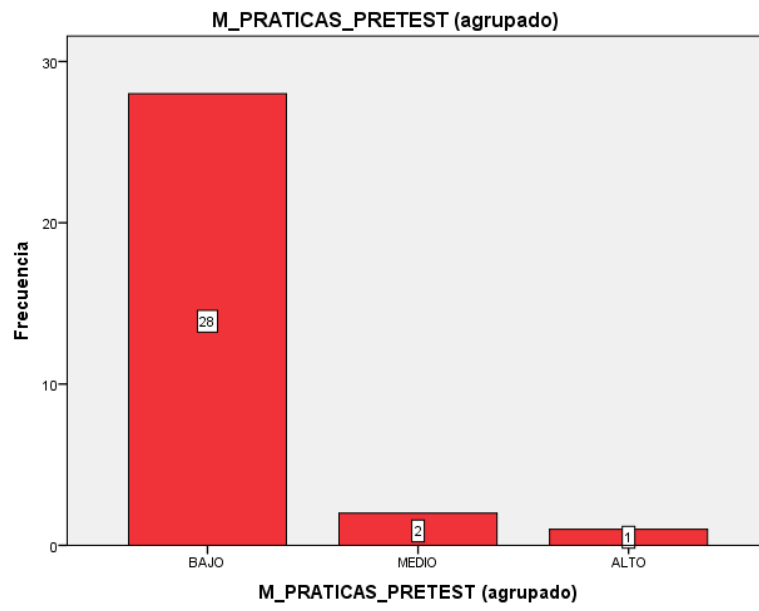


Figura18. Malas prácticas pre test agrupado.

Tabla 53

Frecuencias Malas Prácticas Pre Test

		Frecuenci a	Porcentaj e	Porcentaje válido	Porcentaje acumulado
Válido	0	2	6,5	6,5	6,5
	12	1	3,2	3,2	9,7
	14	2	6,5	6,5	16,1
	16	1	3,2	3,2	19,4
	18	1	3,2	3,2	22,6
	20	3	9,7	9,7	32,3
	22	5	16,1	16,1	48,4
	25	3	9,7	9,7	58,1
	27	2	6,5	6,5	64,5
	28	2	6,5	6,5	71,0
	33	6	19,4	19,4	90,3
	41	1	3,2	3,2	93,5
	50	1	3,2	3,2	96,8
	55	1	3,2	3,2	100,0
	Total	31	100,0	100,0	

La tabla muestra que la mayor frecuencia la tiene la calificación 33 con el 19.4% y 22 con el 16.1% de la variable Malas prácticas en pre test.

3.5.2 Estadísticos variable Malas Prácticas Pretest

Tabla 54

Estadísticos Malas Prácticas Pre Test

<u>M_PRATICAS_PRETEST</u>		
N	Válido	31
	Perdidos	0
Media		24,94
Mediana		25,00
Desviación estándar		11,693

Para la variable M. PRACTICAS_PRETEST, se tienen los siguientes estadígrafos:

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **24.94 puntos**. Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica MALAS PRACTICAS en pre test 24.94.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **25.00**. El 50% de los programas de la muestra tienen una calificación mayor o igual a 25 en la variable MALAS PRACTICAS_PRETEST y 50% tienen menos o igual 25 puntos de calificación en su variable MALAS PRACTICAS_PRETEST en Pre test.

Medidas de Dispersión

La Desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media. Corresponde a **11.693**.

3.5.3 Análisis descriptivo variable Malas Prácticas post test categorizado

Tabla 55

Estadísticos Malas Prácticas Pos Test

M_PRACTICAS_POSTEST		
(agrupado)		
N	Válido	31
	Perdidos	0
Media		1,35
Mediana		1,00
Desviación estándar		,608

Tabla 56

Malas prácticas Pos Test (agrupado)

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	BAJO	22	71,0	71,0	71,0
	MEDIO	7	22,6	22,6	93,5
	ALTO	2	6,5	6,5	100,0
	Total	31	100,0	100,0	

En la tabla se ve el número de programas analizados según la calificación y su porcentaje, 22 programas está con malas prácticas y representa el 71 % , 7 programas tienen calificación medio en malas prácticas y son el 22.5%, y 2 programas tienen calificación alta en M.Practicas con el 6.5% , lo que dice que hay más programas con mejor gestión de malas prácticas en categoría bajo en el post test que en pre test , hay más programas con categoría alta en gestión de malas prácticas en pos test que en pre test. Lo que permite concluir que el uso de QSOURCE mejora la gestión de malas prácticas en aplicaciones desarrolladas en RPG

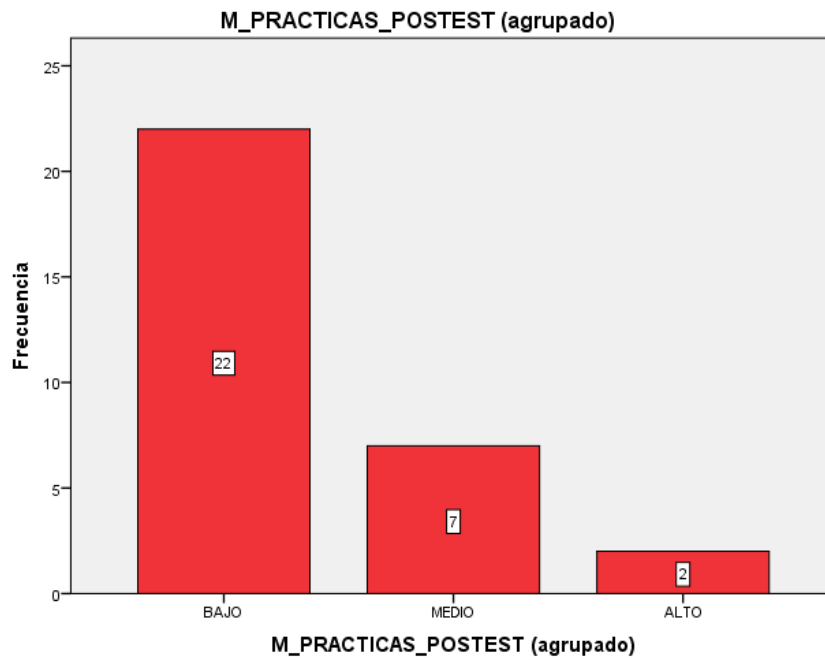


Figura19. Malas prácticas pos test agrupado.

Tabla 57

Frecuencias Malas Prácticas Pos Test

	Frecuencia	Porcentaje	Porcentaje	Porcentaje
	a	e	válido	acumulado
Válido	0	2	6,5	6,5
	16	1	3,2	9,7
	20	5	16,1	25,8
	25	1	3,2	29,0
	28	3	9,7	38,7
	33	8	25,8	64,5
	40	2	6,5	71,0
	42	1	3,2	74,2
	50	6	19,4	93,5
	56	1	3,2	96,8
	71	1	3,2	100,0
Total	31	100,0	100,0	

La tabla muestra que la mayor frecuencia la tiene la calificación 33 con 25% y 50 con 19% en el post test contra 33 con el 19.4% y 22 con el 16.1% de la variable Malas prácticas en pre test. Lo que permite concluir que se ha mejorado la frecuencia de programas RPG con buena gestión de malas prácticas debido al uso de a herramienta QSOURCE.

3.5.4 Estadísticos variable Malas Prácticas pos test

Tabla 58

Estadísticos Malas Prácticas Pos Test

<u>M_PRACTICAS_POSTEST</u>		
N	Válido	31
	Perdidos	0
Media		33,48
Mediana		33,00
Desviación estándar		15,631

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **33.40 puntos.** Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica MALAS PRACTICAS en pos test 33.40 contra 24.94 de pre test, lo que muestra una mejor calificación promedio de los programas en su variable MALAS PRACTICAS en el pos test.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **33.00.** El 50% de los programas de la muestra tienen una calificación mayor o igual a 33 en la variable MALAS PRACTICAS y 50% tienen menos o igual 33 puntos de calificación en su variable MALAS PRACTICAS en pos test contra 25 en el pre test, lo que permite concluir que en el pos

test se ha incrementado el promedio de programas con mejor calificación de MALAS PRACTICAS.

Medidas de Dispersión

La Desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media.

Corresponde a **15.631 en pos test** contra 11.693 del pre test, lo que permite concluir que los valores de la calificación de los programas de la muestra en su variable MALAS PRACTICAS tienen una mayor variabilidad en el pos test debido a los nuevos valores según las mejoras introducidas

3.5.5 Prueba hipótesis variable Malas Prácticas post test Prueba T

Tabla 59

Estadísticos de muestras emparejadas Malas Practicas

		Media	N	Desviación estándar	Media de error estándar
Par 1	M_PRATICAS_PRE	24,94	31	11,693	2,100
	TEST				
	M_PRACTICAS_PO	33,48	31	15,631	2,807
	STEST				

La media para la variable M_PRACTICAS_POSTEST se incrementó en el pos test lo que sugiere una mejora en M_PRACTICAS de los programas de la muestra modificados por la herramienta QSOURCE en el pos test.

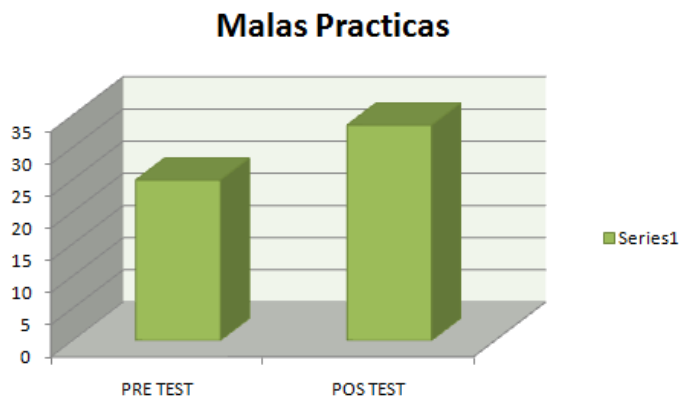


Figura20. Malas prácticas pre y pos test.

Tabla 60

Correlaciones de muestras emparejadas Malas Practicas

		N	Correlación	Sig.
Par 1	M_PRATICAS_PRETEST & M_PRACTICAS_POSTEST	31	,934	,000

Tabla 61

Prueba de muestras emparejadas Malas Practicas

		Diferencias emparejadas							
		95% de intervalo de confianza de la diferencia							
		Media	Desviación estándar	Media de error estándar	Inferior	Superior	t	gl	Sig. (bilateral)
Par 1	M_PRATICAS _PRETEST - M_PRACTICA S_POSTEST	-8,548	6,308	1,133	-10,862	-6,235	-7,545	30	,000

H₀: la aplicación. QSOURCE no mejora significativamente LA GESTION DE MALAS PRACTICAS
H₁: la aplicación. QSOURCE mejora significativamente LA GESTION DE MALAS PRACTICAS

Nivel significación: 0.05 (Max. Nivel error que quiero cometer en mi investigación, Si llevamos a cabo el mismo estudio 100 veces, aproximadamente 95 de los intervalos

de confianza cubriría la verdadera media de la población)

Criterio: si $p\text{valor} < 0.05$ rechazo la H_0 sino lo acepto

Según prueba de T. para muestra relacionada, el sig 0.000 < 0.05 entonces se rechaza H_0 por lo tanto existe fuerte evidencia estadística para afirmar que existe diferencia en la medida de malas prácticas luego de la aplicación de QSOURCE, concluyendo que mejora significativamente la gestión de LAS MALAS PRACTICAS de programas desarrollados en RPG.

3.5.6 Análisis de Normalidad

Para que los test de pruebas emparejadas sean válidos la diferencia entre los valores debe ser distribuida normalmente.

Tabla 62

Resumen de procesamiento de casos Malas Practicas

	Casos					
	Válido		Perdidos		Total	
	N	Porcentaj e	N	Porcentaj e	N	Porcentaj e
DIFF_M_PRACTIC AS	31	100,0%	0	0,0%	31	100,0%

Esta tabla muestra el cuadro de resumen que indica el número y porcentaje de casos analizados, son 31 y corresponde al 100%.

Tabla 63
Pruebas de Normalidad Malas Prácticas

	Kolmogorov-Smirnov ^a			Shapiro-wilk		
	Estadístic		Sig.	Estadístic		Sig.
	o	gl		o	gl	
DIFF_M_PRACTIC AS	,178	31	,014	,918	31	,021

a. Corrección de significación de Lilliefors

Prueba normalidad:

Ho: existe normalidad en la distrib.
H1: no existe normalidad

Alfa = 0.05

Criterio: si pvalor <0.05 se rechaza la Ho sino se acepta

Se toma estadístico shapiro-wilk $n < 50$, pvalor 0.21 >0.05, se acepta Ho, existe normalidad.

DIFF_M_PRACTICAS

DIFF_M_PRACTICAS Stem-and-Leaf Plot

Frequency Stem & Leaf

```

4.00  0 . 0000
2.00  0 . 23
3.00  0 . 455
9.00  0 . 66666677
2.00  0 . 89
3.00  1 . 111
.00   1 .
1.00  1 . 4
6.00  1 . 677777
1.00 Extremes (>=25)

```

Stem width: 10.00
Each leaf: 1 case(s)

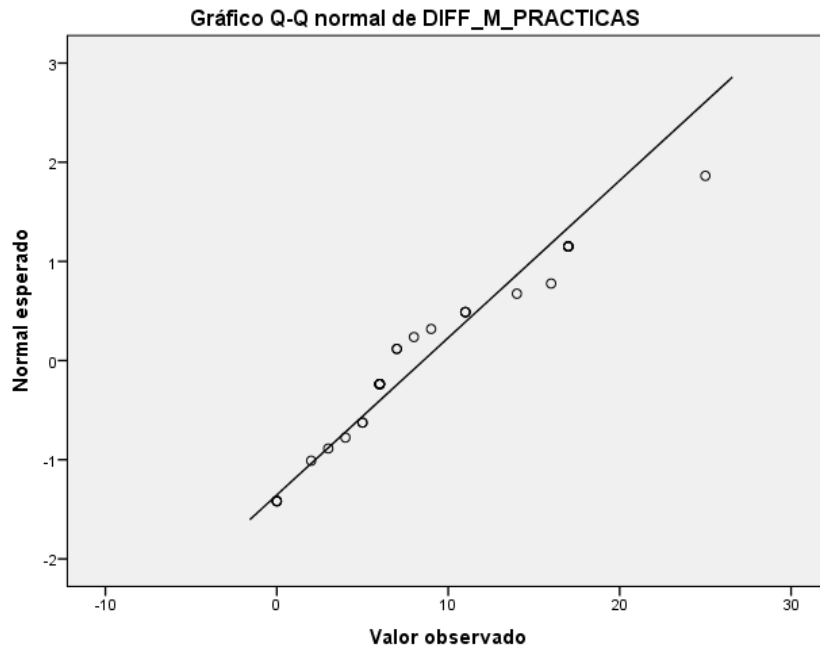


Figura21. Normalidad Malas prácticas.

La línea representa la distribución normal y los puntos la distribución de los datos de la muestra. Se concluye que los datos se comportan conforme a la normal ya que están ubicados sobre la línea (o lo más cercano posible).

3.6 Resultados variable Seguridad

3.6.1 Análisis descriptivo variable Seguridad pre test categorizado

Tabla 64

Estadísticos seguridad pre test agrupado

SEGURIDAD_PRETEST		
(agrupado)		
N	Válido	31
	Perdido	0
<hr/>		
	s	

Tabla 65

Frecuencias Seguridad pre test agrupado

		Frecuencia	Porcentaje	Porcentaje	Porcentaje
		a	e	válido	acumulado
Válido	alto	31	100,0	100,0	100,0

En el gráfico se ve el número de programas analizados según la calificación y su porcentaje, 31 programas está con categoría alto en seguridad y representa el 100, lo que dice que todos los programas de la muestra tienen seguridad alta en el pre test.

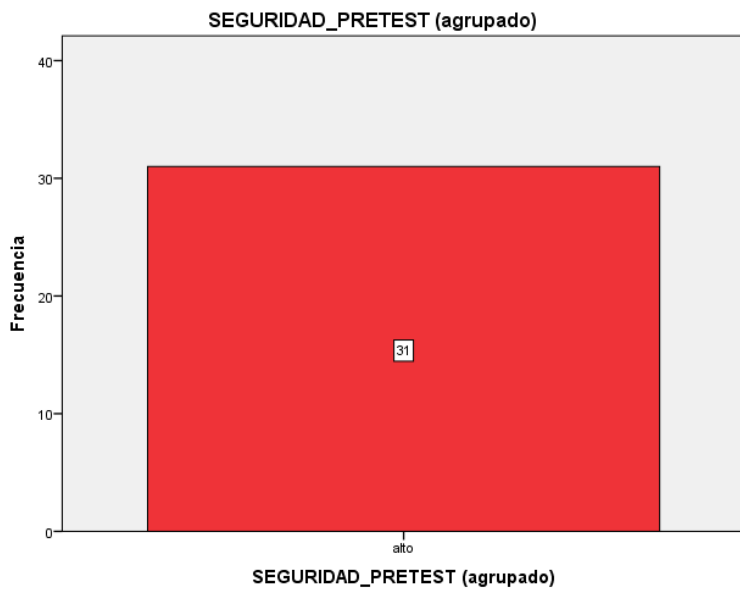


Figura22. Seguridad pre test agrupado.

3.6.2 Estadísticos variable Seguridad pre test agrupado

Tabla 66

Estadísticos Seguridad pre test agrupado

SEGURIDAD_PRETEST_		
AGRUPADO		
N	Válido	31
	Perdido	0
s		
Media		100,00
Mediana		100,00
Desviación		,000
estándar		

Tabla 67

Frecuencia seguridad pre test agrupado

	Frecuen	Porcent	Porcentaje	Porcentaje
	cia	aje	válido	acumulado
Válido	100	31	100,0	100,0

3.6.3 Estadísticos variable Seguridad pre test

Tabla 68.

Estadísticos seguridad pre test

SEGURIDAD_PRETEST		
N	Válido	31
	Perdido	0
s		
Media		100,00
Mediana		100,00
Desviación		,000
estándar		

Para la variable SEGURIDAD, se tienen los siguientes estadígrafos:

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **100.00 puntos**. Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica SEGURIDAD en pre test 100.00.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **100.00**. El 100% de los programas de la muestra tienen una calificación igual a 100 en su variable SEGURIDAD en Pre test.

Medidas de Dispersión

La Desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media. Corresponde a **0.0**.

3.6.4 Análisis descriptivo variable Seguridad pos test categorizado

Tabla 69

Estadísticos seguridad pos test agrupado

SEGURIDAD_POSTEST (agrupado)		
N	Válido	31
	Perdido	0
s		

Tabla 70

Frecuencia seguridad pos test agrupado

		Frecuencia	Porcentaje	Porcentaje	Porcentaje
		a	e	válido	acumulado
Válido	alto	31	100,0	100,0	100,0

En el gráfico se ve el número de programas analizados según la calificación y

su porcentaje, 31 programas está con categoría alto en seguridad y representa el 100 , lo que dice que todos los programas de la muestra tienen seguridad alta en el pos test, igual que en el pre test.

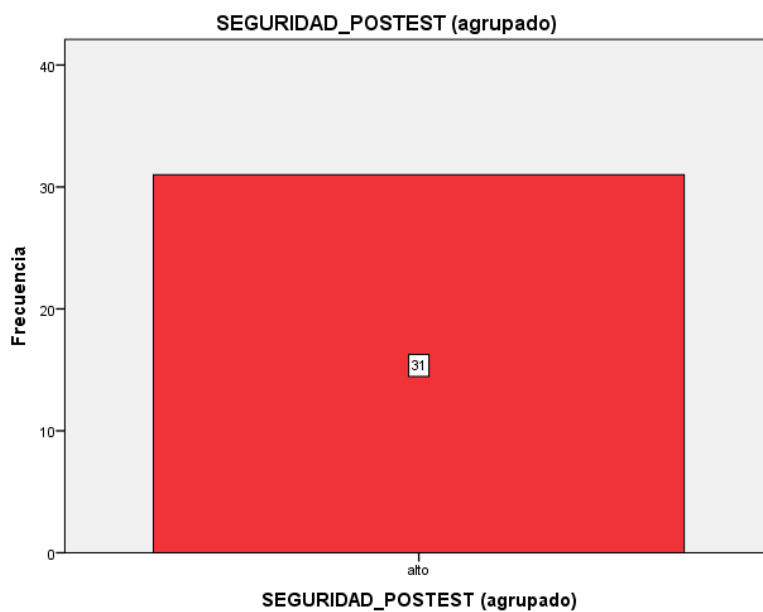


Figura23. Seguridad pos test agrupado.

Tabla 71

Frecuencia seguridad pos test

	Frecuencia	Porcentaje	Porcentaje	Porcentaje
	a	e	válido	acumulado
Válido	100	31	100,0	100,0

3.6.5 Estadísticos variable Seguridad pos test

Tabla 72

Estadísticos seguridad pos test

<u>SEGURIDAD_POSTEST</u>		
N	Válido	31
	Perdido	0
	s	
Media		100,00
Mediana		100,00
Desviación		,000
	estándar	

Para la variable SEGURIDAD, se tienen los siguientes estadígrafos:

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **100.00 puntos**. Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica SEGURIDAD en pos test 100.00.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **100.00**. El 100% de los programas de la muestra tienen una calificación igual a 100 en su variable SEGURIDAD en Pos test.

Medidas de Dispersión

La Desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media. Corresponde a **0.0**.

3.6.6 Prueba hipótesis variable Seguridad post test Prueba T

Advertencias

No se genera la tabla Correlaciones de muestras emparejadas.

No se genera la tabla de prueba de muestras emparejadas.

Tabla 73

Estadísticas muestras emparejadas seguridad

		Media	N	Desviación estándar	Media de error estándar
Par 1	SEGURIDAD_PRET EST	100,00^a	31	,000	,000
	SEGURIDAD_POST EST	100,00^a	31	,000	,000

a. La correlación y t no se pueden calcular porque el error estándar de la diferencia es 0.

La tabla 73 muestra la evidencia de que es invariable el valor 100 en el pre test como en el pos test para la variable SEGURIDAD, para todos los programas de la muestra.

3.6.7 Análisis de Normalidad

Advertencias

diff_seguridad es constante. Se incluirá en cualquier diagrama de caja generada, pero se omitirá otro resultado.

Tabla 74

Resumen casos seguridad

Resumen de procesamiento de casos						
	Válido		Casos Perdidos		Total	
	N	Porcentaje	N	Porcentaje	N	Porcentaje
diff_seguridad	31	100,0%	0	0,0%	31	100,0%

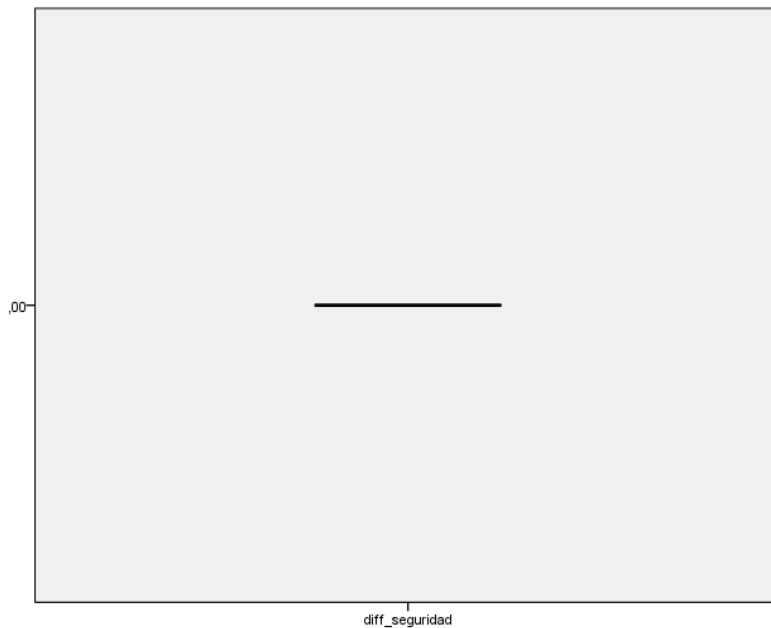


Figura24. Normalidad seguridad.

El diagrama de la figura 24, es el resultado del valor 100 en promedio para todos los programas de la muestra, lo que no es significativo para resultados de comparación.

3.7 Resultados variable complejidad

3.7.1 Análisis descriptivo variable Complejidad pre test categorizado

Tabla 75

Estadísticos Complejidad Pre Test

COMPLEJIDAD_PRETEST (agrupado)		
N	Válido	31
	Perdidos	0
Media		2,52
Mediana		3,00
Desviación estándar		,508

Tabla 76

Frecuencia Complejidad Pre Test (agrupado)

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	MEDIO	15	48,4	48,4	48,4
	ALTO	16	51,6	51,6	100,0
	Total	31	100,0	100,0	

En el gráfico se ve el número de programas analizados según la calificación y su porcentaje, 15 programas está con categoría medio en complejidad y representa el 48.4 %, 16 programas tienen calificación alto en complejidad y son el 51.6%, lo que dice que hay más programas con gestión de complejidad alta en el pre test.

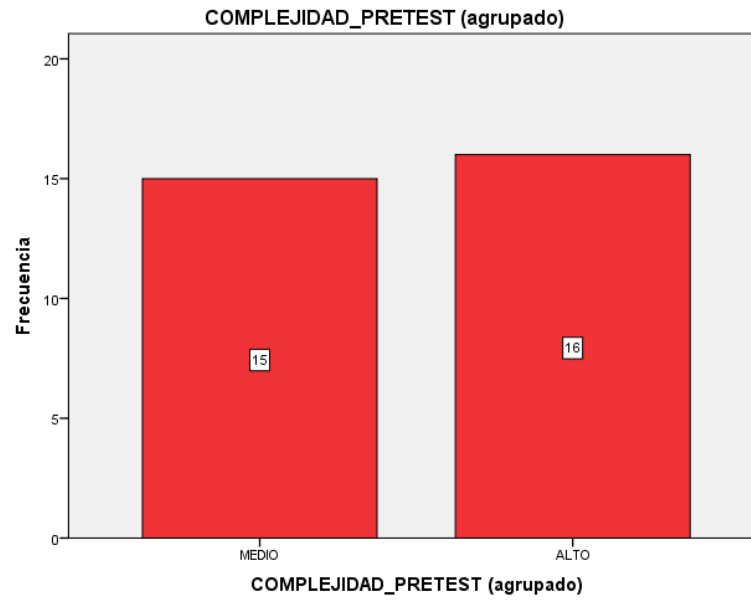


Figura25. Complejidad pre test.

Tabla 77

Frecuencias Complejidad Pre Test

		Frecuenci a	Porcentaj e	Porcentaje válido	Porcentaje acumulado
Válido	41	2	6,5	6,5	6,5
	42	1	3,2	3,2	9,7
	46	8	25,8	25,8	35,5
	50	4	12,9	12,9	48,4
	53	2	6,5	6,5	54,8
	54	1	3,2	3,2	58,1
	55	2	6,5	6,5	64,5
	57	1	3,2	3,2	67,7
	58	1	3,2	3,2	71,0
	63	3	9,7	9,7	80,6
	64	2	6,5	6,5	87,1
	65	1	3,2	3,2	90,3
	66	1	3,2	3,2	93,5
	69	1	3,2	3,2	96,8
	80	1	3,2	3,2	100,0
Total		31	100,0	100,0	

La tabla muestra que a mayor frecuencia la tiene la calificación 46 con el 25% en el pre test para la variable Complejidad en el pre test.

3.7.2 Estadísticos variable Complejidad Pre test

Tabla 78

Estadísticos Complejidad Pre Test

COMPLEJIDAD_PRETEST		
N	Válido	31
	Perdidos	0
Media		54,00
Mediana		53,00
Desviación estándar		9,480

Para la variable COMPLEJIDAD, se tienen los siguientes estadígrafos:

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **54.00 puntos**. Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica COMPLEJIDAD en pre test 54.00.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **53.00**. El 50% de los programas de la muestra tienen una calificación mayor o igual a 53 en la variable COMPLEJIDAD y 50% tienen menos o igual 53 puntos de calificación en su variable COMPLEJIDAD en Pre test.

Medidas de Dispersión

La Desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media. Corresponde a **9.480**.

3.7.3 Análisis descriptivo variable Complejidad post test categorizado

Tabla 79

Estadísticos Complejidad Pos Test

COMPLEJIDAD_POSTEST		
(agrupado)		
N	Válido	31
	Perdidos	0
Media		2,94
Mediana		3,00
Desviación estándar		,250

Tabla 80

Frecuencia Complejidad Pos Test (agrupado)

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	MEDIO	2	6,5	6,5	6,5
	ALTO	29	93,5	93,5	100,0
	Total	31	100,0	100,0	

En la tabla se ve el número de programas analizados según la calificación y su porcentaje, 2 programas está con categoría medio en complejidad y representa el 6.5% , 29 programas tienen calificación alto en complejidad y son el 93.5%, lo que dice que hay más programas con complejidad alta en el pos test que en pre test. Lo que permite concluir que el uso de QSOURCE mejora la complejidad de aplicaciones desarrolladas en RPG

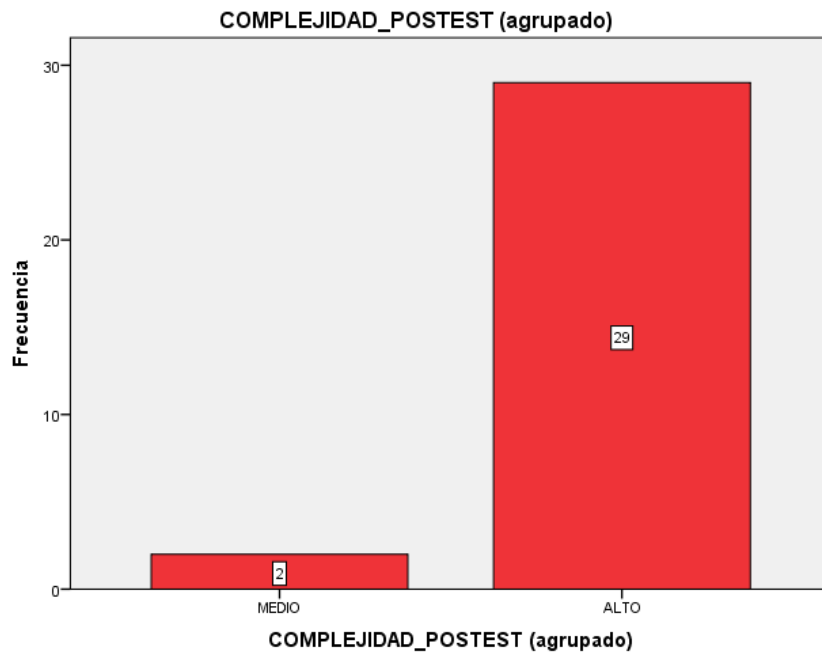


Figura26. Complejidad pos test.

Tabla 81

Frecuencias Complejidad Pos Test

		Frecuenci a	Porcentaj e	Porcentaje válido	Porcentaje acumulado
Válido	50	2	6,5	6,5	6,5
	53	4	12,9	12,9	19,4
	57	2	6,5	6,5	25,8
	58	1	3,2	3,2	29,0
	61	5	16,1	16,1	45,2
	62	1	3,2	3,2	48,4
	63	1	3,2	3,2	51,6
	64	2	6,5	6,5	58,1
	66	2	6,5	6,5	64,5
	69	2	6,5	6,5	71,0
	70	2	6,5	6,5	77,4
	71	3	9,7	9,7	87,1
	73	1	3,2	3,2	90,3
	77	1	3,2	3,2	93,5
	80	2	6,5	6,5	100,0
Total		31	100,0	100,0	

La tabla muestra que a mayor frecuencia la tiene la calificación 61 con el 16.1% en pos test contra 46 con el 25% en el pre test para la variable Complejidad, lo que permite concluir que en pos test se ha mejorado la frecuencia de programas con mejor calificación de la variable CALIDAD, fruto de los cambios generados por la herramienta QSOURCE.

3.7.4 Estadísticos variable Complejidad pos test

Tabla 82

Estadísticos Complejidad Pos Test

<u>COMPLEJIDAD_POSTEST</u>		
N	Válido	31
	Perdidos	0
Media		63,71
Mediana		63,00
Desviación		8,359
estándar		

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **63.71 puntos.** Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica COMPLEJIDAD en pos test 63.71 contra 54.00 de pre test, lo que muestra una mejor calificación promedio de los programas en su variable COMPLEJIDAD en el pos test.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **63.00.** El 50% de los programas de la muestra tienen una calificación mayor o igual a 63 en la variable COMPLEJIDAD y 50% tienen menos o igual 63 puntos de calificación en su variable COMPLEJIDAD en pos test contra 53 en el pre test, lo que permite concluir que en el pos test se ha incrementado el promedio de programas con mejor calificación de COMPLEJIDAD.

Medidas de Dispersión

La Desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media. Corresponde a **8.359 en pos test** contra 9.480 del pre test, lo que permite concluir que los valores de la calificación de los programas de la muestra en su COMPLEJIDAD

tienen una mayor variabilidad en el pre test debido a que los cambios no generaron mejores valores.

3.75 Prueba hipótesis variable Complejidad post test Prueba T

Tabla 83

Estadísticas de muestras emparejadas Complejidad

		Media	N	Desviación estándar	Media de error estándar
Par 1	COMPLEJIDAD_PR E TEST	54,00	31	9,480	1,703
	COMPLEJIDAD_PO S TEST	63,71	31	8,359	1,501

La media para la variable COMPLEJIDAD_POSTEST se incrementó en el pos test lo que sugiere una mejora en el COMPLEJIDAD de los programas de la muestra modificados por la herramienta QSOURCE en el pos test.

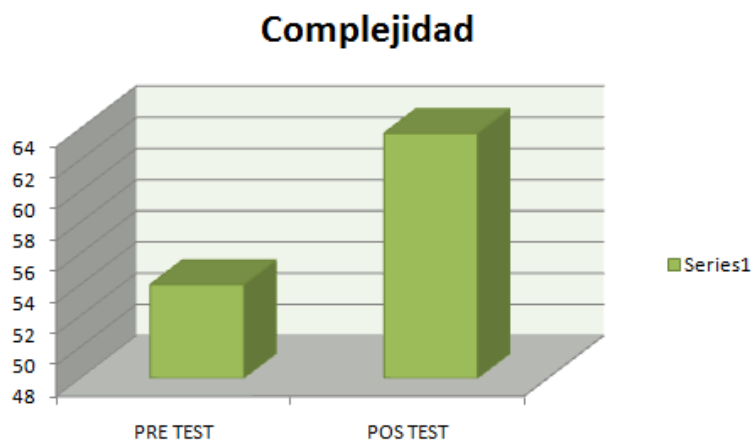


Figura27. Complejidad pre y pos test.

Tabla 84

Correlaciones muestras emparejadas Complejidad

		Correlación		
		N	n	Sig.
Par 1	COMPLEJIDAD_PR E TEST & COMPLEJIDAD_PO S TEST	31	,836	,000

Tabla 85

Prueba de muestras emparejadas Complejidad

		Diferencias emparejadas							
		95% de intervalo de confianza							Sig.
		Media	Desviación estándar	Media de error estándar	de la diferencia		t	gl	(bilateral)
					Inferior	Superior			
Par 1	COMPLEJIDAD _PRETEST - COMPLEJIDAD _POSTEST	-9,710	5,217	,937	-11,623	-7,796	-10,363	30	,000

Ho: la aplicación. QSOURCE no mejora significativamente LA COMPLEJIDAD
H1: la aplicación. QSOURCE mejora significativamente LA COMPLEJIDAD

Nivel significación: 0.05 (Max. Nivel error que quiero cometer en mi investigación, Si llevamos a cabo el mismo estudio 100 veces, aproximadamente 95 de los intervalos de confianza cubriría la verdadera media de la población)

Criterio: si $p\text{valor} < 0.05$ rechazo la H_0 sino lo acepto. Según prueba de T. para muestra relacionada, el sig 0.000 < 0.05 entonces se rechaza H_0 por lo tanto existe fuerte evidencia estadística para afirmar existe diferencia en la complejidad luego de la aplicación de QSOURCE, concluyendo que mejora significativamente la gestión de

COMPLEJIDAD de programas desarrollados en RPG.

3.7.6 Análisis de Normalidad

Para que los test de pruebas emparejadas sean válidos la diferencia entre los valores debe ser distribuida normalmente.

Tabla 86

Resumen de procesamiento de casos Complejidad

	Casos					
	Válido		Perdidos		Total	
	N	Porcentaje	N	Porcentaje	N	Porcentaje
DIFF_COMPLEJIDAD AD	31	100,0%	0	0,0%	31	100,0%

Esta tabla muestra el cuadro de resumen que indica el número y porcentaje de casos analizados, son 31 y corresponde al 100%.

Tabla 87

Prueba de Normalidad Complejidad

	Kolmogorov-Smirnov ^a			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
DIFF_COMPLEJIDAD	,140	31	,123	,956	31	,229

a. Corrección de significación de Lilliefors

Prueba normalidad:

Ho: existe normalidad en la distrib.
H1: no existenormalidad

Alfa = 0.05

Criterio: si pvalor <0.05 se rechaza la Ho sino se acepta

Se toma estadístico shapiro-wilk n < 50, pvalor 0.229 >0.05, se acepta Ho, existe normalidad.

DIFF_COMPLEJIDAD

DIFF_COMPLEJIDAD Stem-and-Leaf Plot

Frequency Stem & Leaf

3.00	0 . 000
1.00	0 . 2
.00	0 .
8.00	0 . 67777777
4.00	0 . 8999
5.00	1 . 11111
3.00	1 . 222
2.00	1 . 55
3.00	1 . 667
1.00	1 . 9
1.00	Extremes (>=20)

Stem width: 10.00

Each leaf: 1 case(s)

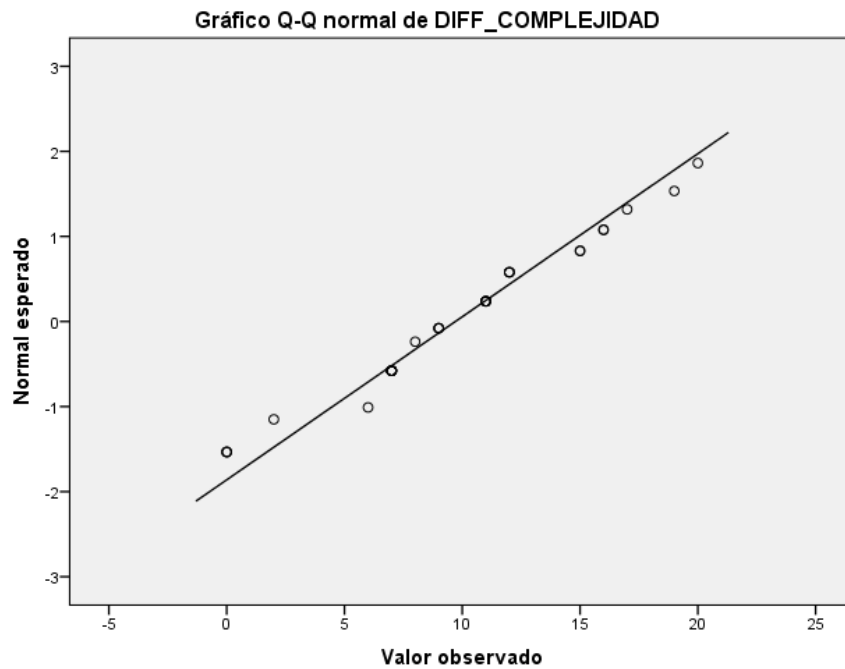


Figura28. Normalidad Complejidad.

La línea representa la distribución normal y los puntos la distribución de los datos de la muestra. Se concluye que los datos se comportan conforme a la normal ya que están ubicados sobre la línea (o lo más cercano posible).

3.8 Resultados variable Obsolescencia

3.8.1 Análisis descriptivo variable Obsolescencia Pre test Categorizado

Tabla 88

Estadísticos Obsolescencia pre test

OBSOLESCENCIA_PRETES		
T (agrupado)		
N	Válido	31
	Perdidos	0
Media		2,10
Mediana		2,00
Moda		2
Desviación estándar		,396

Tabla 89

Obsolescencia pre test agrupado

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válido	bajo	1	3,2	3,2	3,2
	medio	26	83,9	83,9	87,1
	alto	4	12,9	12,9	100,0
	Total	31	100,0	100,0	

En la tabla se ve el número de programas analizados según la calificación y su porcentaje, 1 programa está con categoría bajo en Obsolescencia y representa el

3.2%, 26 programas tienen calificación medio en Obsolescencia y son el 83.9%, y hay 4 programas en calificación alto y son el 12.9%, lo que dice que hay más programas con Obsolescencia media en el pre test.

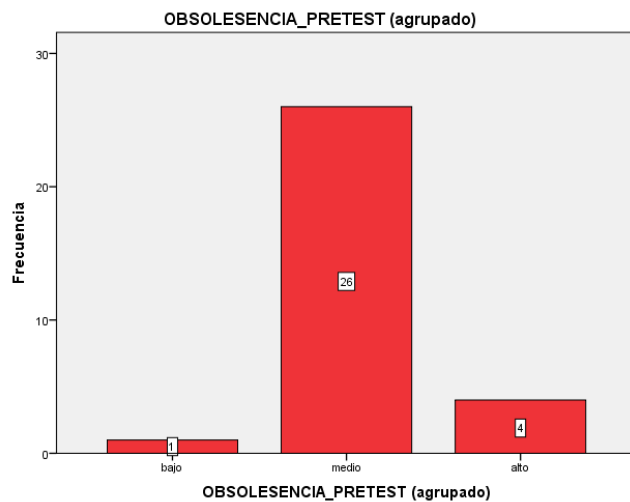


Figura29. Obsolescencia pre test agrupado.

Tabla 90

Obsolescencia pre test

	Frecuencia	Porcentaje	Porcentaje	Porcentaje
	a	e	válido	acumulado
Válido	0	1	3,2	3,2
	50	4	12,9	16,1
	54	1	3,2	19,4
	56	2	6,5	25,8
	57	6	19,4	45,2
	59	1	3,2	48,4
	60	2	6,5	54,8
	63	1	3,2	58,1
	65	5	16,1	74,2
	68	1	3,2	77,4
	70	3	9,7	87,1
	83	3	9,7	96,8
	90	1	3,2	100,0
Total	31	100,0	100,0	

La tabla muestra que a mayor frecuencia la tiene la calificación 57 con el 19.4% en el pre test para la variable Obsolescencia en el pre test.

3.8.2 Estadísticos variable Obsolescencia pre test

Tabla 91

Estadísticos Obsolescencia pre test

OBSOLESCENCIA_PRET		
EST		
N	Válido	31
	Perdido	0
s		
Media		61,03
Mediana		60,00
Moda		57
Desviación estándar		15,357

Para la variable Obsolescencia, se tienen los siguientes estadígrafos:

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **61.03 puntos**. Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica OBSOLESCENCIA en pre test 61.03.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **60.00**. El 50% de los programas de la muestra tienen una calificación mayor o igual a 53 en la variable OBSOLESCENCIA y 50% tienen menos o igual 60 puntos de calificación en su variable OBSOLESCENCIA en Pre test.

Medidas de Dispersión

La Desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media. Corresponde a **15.357**.

3.8.3 Análisis descriptivo variable Obsolescencia post test categorizado

Tabla 92

Estadísticos pos test agrupado

OBSOLESCENCIA_POSTES		
T (agrupado)		
N	Válido	31
	Perdidos	0
Media		2,45
Mediana		2,00
Moda		2ª
Desviación estándar		,568

a. Existen múltiples modos.

Se muestra el valor más
pequeño.

Tabla 93

Obsolescencia pos test agrupado

	Frecuenci a	Porcentaj e	Porcentaje válido	Porcentaje acumulado
Válido bajo	1	3,2	3,2	3,2
medio	15	48,4	48,4	51,6
alto	15	48,4	48,4	100,0
Total	31	100,0	100,0	

En la tabla se ve el número de programas analizados según la calificación y su porcentaje, 2 programas están con categoría bajo en Obsolescencia y representa el 3.2%, 15 programas tienen calificación medio en Obsolescencia y son el 48.4%, y 15 programas tienen calificación alta y son el 48.4% en pos test contra 4 del pre test, lo que dice que hay más programas con complejidad alta en el pos test que en pre test. Lo que permite concluir que el uso de QSOURCE mejora la gestión de la Obsolescencia de aplicaciones desarrolladas en RPG

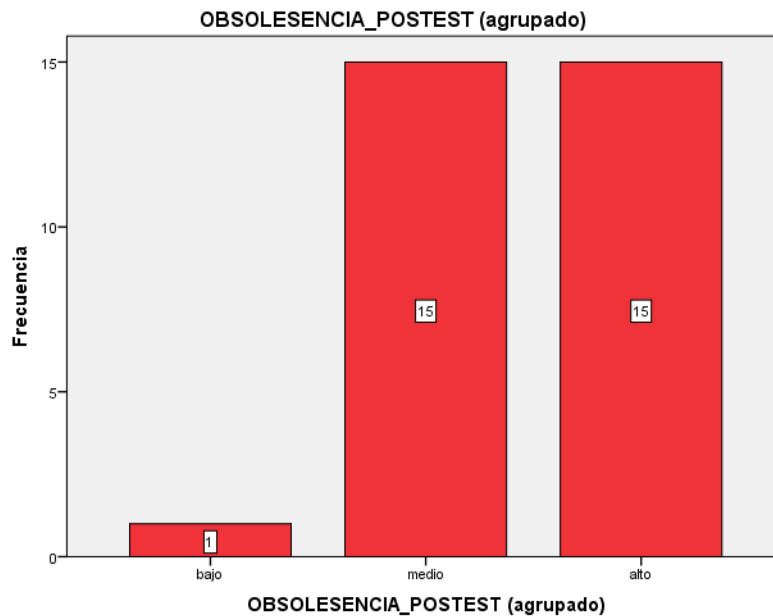


Figura30. Obsolescencia pos test agrupado.

3.8.4 Estadísticos variable Obsolescencia pos test

Tabla 94

Estadísticos Obsolescencia pos test

OBSOLESCENCIA_POST		
EST		
N	Válido	31
	Perdidos	0
	Media	68,06
	Mediana	70,00
	Moda	80
	Desviación estándar	14,654

La media (promedio aritmético) de la distribución es **68.06 puntos**. Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica OBSOLESCENCIA en pos test 68.06 contra 61.03 de pre test, lo

que muestra una mejor calificación promedio de los programas en su variable OBSOLESCENCIA en el pos test.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **70.00**. El 50% de los programas de la muestra tienen una calificación mayor a 70 en la variable OBSOLESCENCIA y 50% tienen menos o igual 70 puntos de calificación en su variable OBSOLESCENCIA en pos test contra 60 en el pre test, lo que permite concluir que en el pos test se ha incrementado el promedio de programas con mejor calificación de OBSOLESCENCIA.

Medidas de Dispersión

La desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media. Corresponde a **14.654 en pos test** contra 15.357 del pre test, lo que permite concluir que los valores de la calificación de los programas de la muestra en su OBSOLESCENCIA tienen una mayor variabilidad en el pre test debido a que los cambios no generaron mejores valores.

3.8.5 Prueba hipótesis variable Complejidad post test Prueba T

Tabla 95

Estadísticas muestras emparejadas

		Media	N	Desviación estándar	Media de error estándar
Par 1	OBSOLESCENCIA_ PRETEST	61,03	31	15,357	2,758
	OBSOLESCENCIA_ POSTEST	68,06	31	14,654	2,632

La media para la variable OBSOLESCENCIA_POSTEST se incrementó en el pos test lo que sugiere una mejora en el OBSOLESCENCIA de los programas de la

muestra modificados por la herramienta QSOURCE en el pos test.

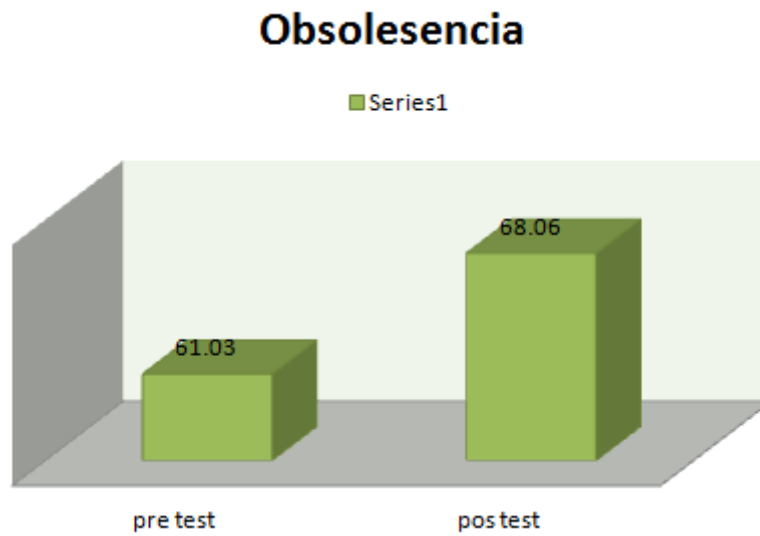


Figura31. Obsolescencia pre y post.

Tabla 96

Correlación muestras emparejadas Obsolescencia

		Correlació		
		N	n	Sig.
Par 1	OBSOLESCENCIA_ PRETEST &OBSOLESCENCIA _POSTEST	31	,704	,000

Tabla 97

Prueba muestras emparejadas Obsolescencia

		Diferencias emparejadas							
		95% de intervalo de confianza							
		Media	Desviación estándar	Media de error estándar	de la diferencia		t	gl	Sig. (bilateral)
					Inferior	Superior			
Par 1	OBSOLESCEN CIA_PRETEST	-7,032	11,563	2,077	-11,274	-2,791	-3,386	30	,002
	- OBSOLESCEN CIA_POSTEST								

Ho: la aplicación. QSOURCE no mejora significativamente LA OBSOLESCENCIA
H1: la aplicación. QSOURCE mejora significativamente LA OBSOLESCENCIA

Nivel significación: 0.05 (Max. Nivel error que quiero cometer en mi investigación, Si llevamos a cabo el mismo estudio 100 veces, aproximadamente 95 de los intervalos de confianza cubriría la verdadera media de la población)

Criterio: si $p\text{valor} < 0.05$ rechazo la H_0 sino lo acepto

Según prueba de T. para muestra relacionada, el sig $0.002 < 0.05$ entonces se rechaza H_0 por lo tanto existe fuerte evidencia estadística para afirmar que existe diferencia en la medida de la obsolescencia luego de la aplicación de QSOURCE concluyendo que mejora significativamente la gestión de OBSOLESCENCIA de programas desarrollados en RPG.

3.8.6 Análisis de Normalidad

Tabla 98

Resumen casos normalidad Obsolescencia

	Casos					
	Válido		Perdidos		Total	
	N	Porcent aje	N	Porcent aje	N	Porcent aje
diff_obsolescencia	31	100,0%	0	0,0%	31	100,0%

Esta tabla muestra el cuadro de resumen que indica el número y porcentaje de casos analizados, son 31 y corresponde al 100%.

Tabla 99

Pruebas normalidad Obsolescencia

	Kolmogorov-Smirnov ^a			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
diff_obsolescencia	,235	31	,0169	,825	31	,0229

a. Corrección de significación de Lilliefors

Prueba normalidad:

Ho: existe normalidad en la distrib.
H1: no existenormalidad

Alfa = 0.05

Criterio: si pvalor <0.05 se rechaza la Ho sino se acepta

Se toma estadístico shapiro-wilk n < 50, pvalor 0.229 >0.05, se acepta Ho, existe normalidad.

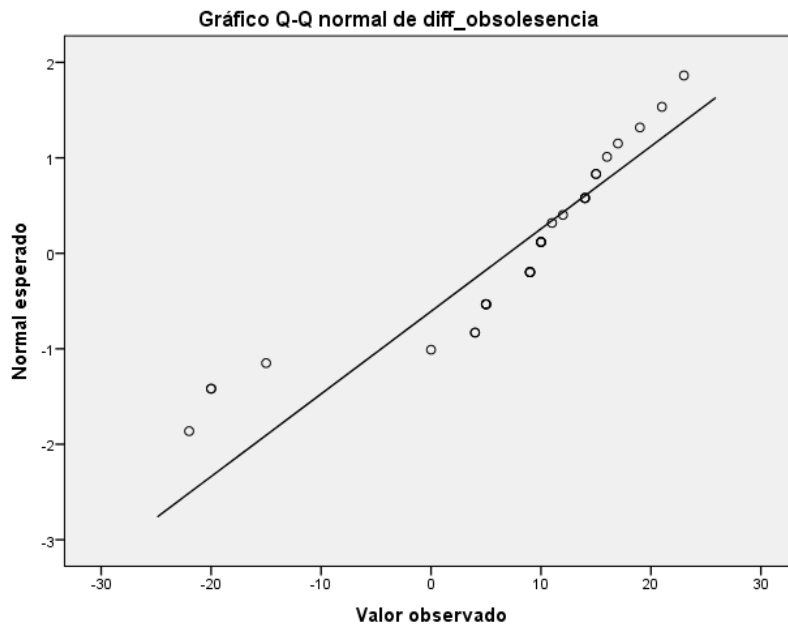


Figura32. Normalidad Obsolescencia.

La línea representa la distribución normal y los puntos la distribución de los datos de la muestra. Se concluye que los datos se comportan conforme a la normal ya que están ubicados sobre la línea (o lo más cercano posible).

3.9 Resultados variable Mantenibilidad

3.9.1 Análisis descriptivo variable Mantenibilidad Pretest categorizado

Tabla 100

Calidad pre test (agrupado)

Mantenibilidad PRETEST (agrupado)		
N	Válido	31
	Perdidos	0
Media		1,06
Mediana		1,00
Desviación estándar		,250

Tabla 101

Mantenibilidad Pre Test (agrupado)

		Frecuenci a	Porcentaj e	Porcentaje válido	Porcentaje acumulado
Válido	BAJO	29	93,5	93,5	93,5
	MEDIO	2	6,5	6,5	100,0
Total		31	100,0	100,0	

En la tabla se ve el número de programas analizados según la calificación y su porcentaje, 29 programas está con baja Mantenibilidad y representa el 93.5%, 2 programas tienen calificación medio en Mantenibilidad y son el 6.5%, lo que dice que hay más programas con una Mantenibilidad bajo en el pre test.

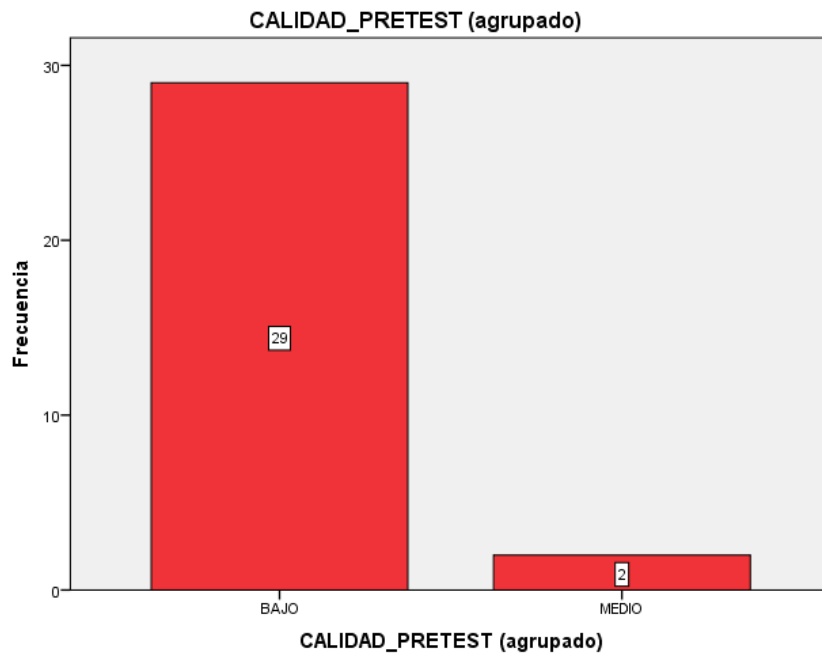


Figura33. Mantenibilidad pre test agrupado.

Tabla 102

Mantenibilidad *Pre Test*

		Frecuenci a	Porcentaj e	Porcentaje válido	Porcentaje acumulado
Válido	18	1	3,2	3,2	3,2
	21	1	3,2	3,2	6,5
	22	2	6,5	6,5	12,9
	23	5	16,1	16,1	29,0
	25	1	3,2	3,2	32,3
	26	1	3,2	3,2	35,5
	27	1	3,2	3,2	38,7
	28	1	3,2	3,2	41,9
	29	3	9,7	9,7	51,6
	30	1	3,2	3,2	54,8
	31	2	6,5	6,5	61,3
	33	4	12,9	12,9	74,2
	35	1	3,2	3,2	77,4
	38	2	6,5	6,5	83,9
	40	2	6,5	6,5	90,3
	41	1	3,2	3,2	93,5
	45	1	3,2	3,2	96,8
	50	1	3,2	3,2	100,0
Total		31	100,0	100,0	

En esta tabla se observa que la frecuencia más alta la tiene la puntuación 23 de Mantenibilidad en pre test con el 16%. En pre test.

3.9.2 Estadísticos variable Mantenibilidad Pretest

Tabla 103

Estadísticos Mantenibilidad Pre Test

<u>MANTENIBILIDAD_PRETEST</u>		
N	Válido	31
	Perdidos	0
Media		30,39
Mediana		29,00
Desviación estándar		7,719

Para la variable MANTENIBILIDAD_PRETEST, se tienen los siguientes estadígrafos:

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **30.39 puntos**. Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica MANTENIBILIDAD en pre test 30.39.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **29.00**. El 50% de los programas de la muestra tienen una calificación mayor o igual a 29 en la variable MANTENIBILIDAD_PRETEST y 50% tienen menos o igual a 29 puntos de calificación en su variable MANTENIBILIDAD_PRETEST en Pre test.

Medidas de Dispersión

La Desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media. Corresponde a **7.719**.

3.9.3 Análisis descriptivo variable Mantenibilidad post testCategorizado

Tabla 104

Mantenibilidad *Pos Test* (agrupado)

MANTENIBILIDAD_POSTEST (agrupado)		
N	Válido	31
	Perdidos	0
Media		2,71
Mediana		3,00
Desviación estándar		,529

Tabla 105

Mantenibilidad *Pos Test* (agrupado)

		Frecuenci a	Porcentaj e	Porcentaje válido	Porcentaje acumulado
Válido	BAJO	1	3,2	3,2	3,2
	MEDIO	7	22,6	22,6	25,8
	ALTO	23	74,2	74,2	100,0
	Total	31	100,0	100,0	

En la tabla se ve el número de programas analizados según la calificación y su porcentaje, 1 programa está con baja Mantenibilidad y representa el 3.2%, 7 programas tienen calificación medio en Mantenibilidad y son el 22.6%, y 23 programas tienen una calificación alta con 74% contra el 93% de bajo en pre test, lo que dice que hay más programas con un Mantenibilidad alto en el pos test que en el pre test. Lo que permite concluir que el uso de QSOURCE mejora la Mantenibilidad de aplicaciones desarrolladas en RPG

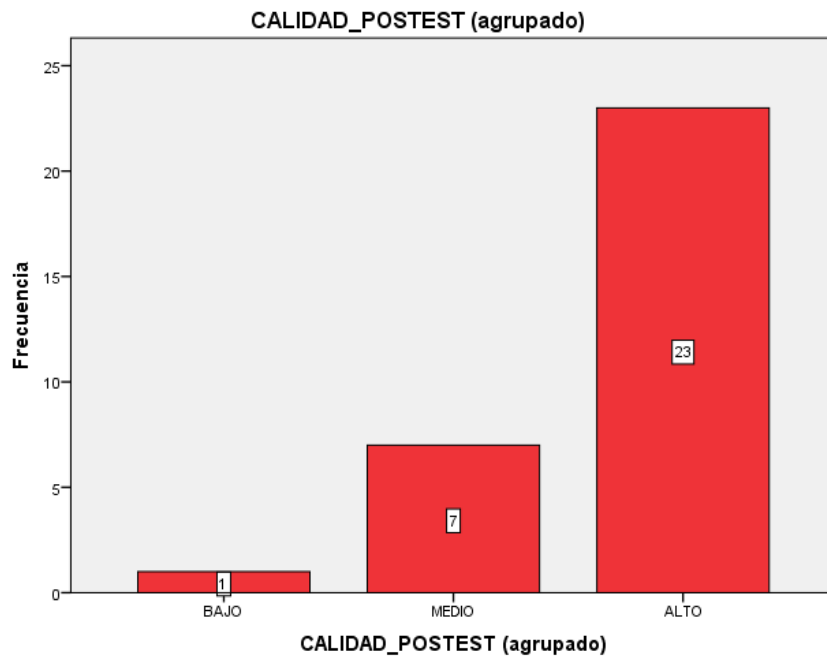


Figura34. Mantenibilidad pos test agrupado.

Tabla 106

Mantenibilidad *Pos Test*

	Frecuenci a	Porcentaj e	Porcentaje válido	Porcentaje acumulado
Válido	37	1	3,2	3,2
	46	1	3,2	6,5
	48	3	9,7	16,1
	50	3	9,7	25,8
	52	3	9,7	35,5
	54	2	6,5	41,9
	56	1	3,2	45,2
	57	2	6,5	51,6
	60	3	9,7	61,3
	63	4	12,9	74,2
	64	1	3,2	77,4
	65	2	6,5	83,9
	66	2	6,5	90,3
	71	1	3,2	93,5
	75	1	3,2	96,8
	87	1	3,2	100,0
Total	31	100,0	100,0	

En esta tabla se observa que la frecuencia más alta la tiene la puntuación 63 de Mantenibilidad en pos test con el 12.9% contra 23 punto de calidad en pre test con el 16%. Esto lleva a concluir que en pos test luego de aplicar la herramienta QSOURCE se ha incrementado la frecuencia de programas con mejor calificación en la variable Mantenibilidad.

3.9.4 Estadísticos variable Mantenibilidad post test

Tabla 107

Estadísticos Mantenibilidad Pos Test

<u>MANTENIBILIDAD_POSTEST</u>		
N	Válido	31
	Perdidos	0
Media		58,13
Mediana		57,00
Desviación estándar		9,824

Medidas de Tendencia Central

La media (promedio aritmético) de la distribución es **58.13 puntos.** Lo que permite concluir que en promedio los programas de la muestra tienen como calificación de su característica MANTENIBILIDAD en pos test 58.13 contra 30.39 de pre test, lo que muestra una mejor calificación promedio de los programas en su variable MANTENIBILIDAD en el pos test.

La mediana (Valor por encima y por debajo del cual se encuentran la mitad de los casos; o valor central de la distribución es **57.00.** El 50% de los programas de la muestra tienen una calificación mayor o igual a 57 en la variable MANTENIBILIDAD y 50% tienen menos o igual 57 puntos de calificación en su variable MANTENIBILIDAD en pos test contra 29 en el pre test, lo que permite concluir que en el pos test se ha incrementado el promedio de programas con mejor calificación de MANTENIBILIDAD.

Medidas de Dispersión

La Desviación estándar (Medida de dispersión en torno a la media. Raíz cuadrada de la varianza. Mide el grado en que las puntuaciones de la variable se alejan de su media. Corresponde a **9.824 en pos test** contra 7.719 del pre test, lo que permite concluir que los valores de la calificación de los programas de la muestra en su variable MANTENIBILIDAD tienen una mayor variabilidad en el pos test debido a los nuevos valores según las mejoras introducidas.

3.9.5 Prueba de hipótesis variable Mantenibilidad post test Prueba T

Tabla 108

Estadísticas de muestras emparejadas Mantenibilidad

	Media	N	Desviación estándar	Media de error estándar
Par 1 MANTENIBILIDAD_ PRETEST	30,39	31	7,719	1,386
MANTENIBILIDAD_ POSTEST	58,13	31	9,824	1,764

La media para la variable MANTENIBILIDAD_POSTEST se incrementó en el pos test lo que sugiere una mejora en el MANTENIBILIDAD de los programas de la muestra modificados por la herramienta QSOURCE en el pos test.

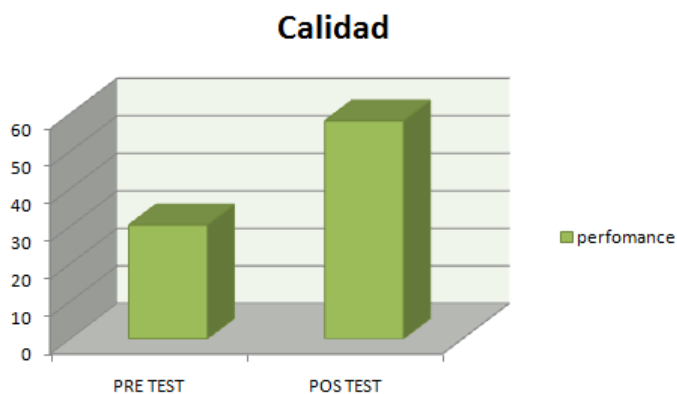


Figura35. Calidad pre y pos test.

Tabla 109

Correlación de muestras emparejadas Mantenibilidad

		N	Correlación	Sig.
Par 1	MANTENIBILIDAD_PRETEST &MANTENIBILIDAD_POSTE ST	31	,687	,000

Tabla 110

Prueba de muestras emparejadas MANTENIBILIDAD

		Diferencias emparejadas							
		95% de intervalo de confianza de							
		Desviación		Media de error	la diferencia		t	gl	Sig.
		Media	estándar	estándar	Inferior	Superior			(bilateral)
Par 1	MANTENIBILIDAD	-	7,206	1,294	-30,385	-25,099	-	30	,000
	_PRETEST -	27,7						21,4	
	MANTENIBILIDAD	42						34	
	_POSTEST								

<p>Ho: la aplicación. QSOURCE no mejora significativamente LA MANTENIBILIDAD</p> <p>H1: la aplicación. QSOURCE mejora significativamente LA MANTENIBILIDAD</p>
--

Nivel significación: 0.05 (Max. Nivel error que quiero cometer en mi investigación, Si llevamos a cabo el mismo estudio 100 veces, aproximadamente 95 de los intervalos de confianza cubriría la verdadera media de la población)

Criterio: si $p\text{valor} < 0.05$ rechazo la H_0 sino lo acepto

Según prueba de T. para muestra relacionada, el sig. 0.000 < 0.05 entonces se rechaza H_0 por lo tanto existe fuerte evidencia estadística para afirmar que existe diferencia en la Mantenibilidad luego de la aplicación de QSOURCE, concluyendo

Que mejora significativamente la Mantenibilidad de programas desarrollados en RPG.

3.9.6 Análisis de normalidad

Para que los test de pruebas emparejadas sean válidos la diferencia entre los valores debe ser distribuida normalmente.

Tabla 111

Resumen de procesamiento de casos MANTENIBILIDAD

	Casos					
	Válido		Perdidos		Total	
	N	Porcentaj e	N	Porcentaj e	N	Porcentaj e
DIFF_MANTENIBILIDAD	31	100,0%	0	0,0%	31	100,0%

Esta tabla muestra el cuadro de resumen que indica el número y porcentaje de casos analizados, son 31 y corresponde al 100%.

Tabla 112

Pruebas de Normalidad MANTENIBILIDAD

	Kolmogorov-Smirnov ^a			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
DIFF_MANTENIBILIDA D	,116	31	,200 [*]	,955	31	,219

*. Esto es un límite inferior de la significación verdadera.

a. Corrección de significación de Lilliefors

Prueba normalidad:

Ho: existe normalidad en la distrib.
H1: no existenormalidad

Alfa = 0.05

Criterio: si pvalor <0.05 se rechaza la Ho sino se acepta

Se toma estadístico shapiro-wilk $n < 50$, pvalor 0.219 >0.05, se acepta Ho, existe normalidad.

DIFF_MANTENIBILIDAD

DIFF_MANTENIBILIDAD Stem-and-Leaf Plot

Frequency Stem & Leaf

1.00 Extremes (= <7)

.00 1 .

1.00 1 . 5

7.00 2 . 1223344

10.00 2 . 5556667799

7.00 3 . 0111123

3.00 3 . 779

1.00 4 . 0

1.00 Extremes (>=42)

Stem width: 10.00

Each leaf: 1 case(s)

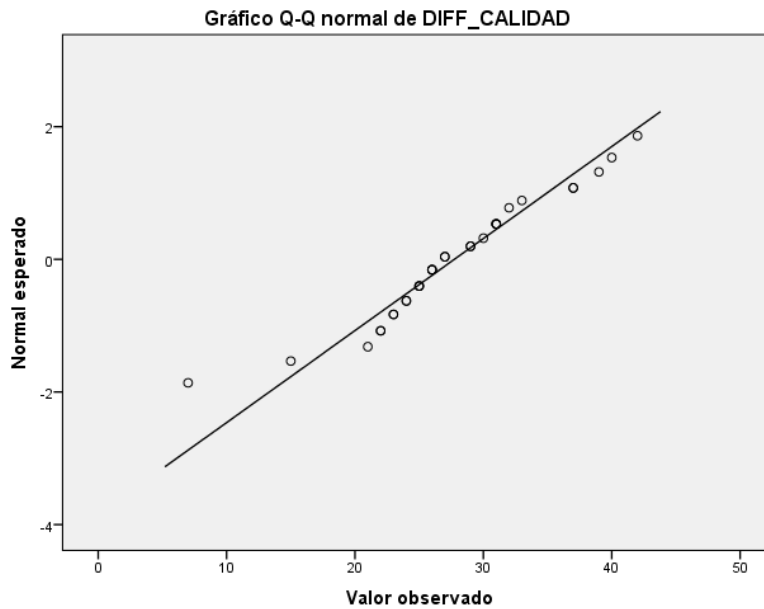


Figura36. Mantenibilidad de Calidad.

La línea representa la distribución normal y los puntos la distribución de los datos de la muestra. Se concluye que los datos se comportan conforme a la normal ya que están ubicados sobre la línea (o lo más cercano posible).

3.10 Resultados generales

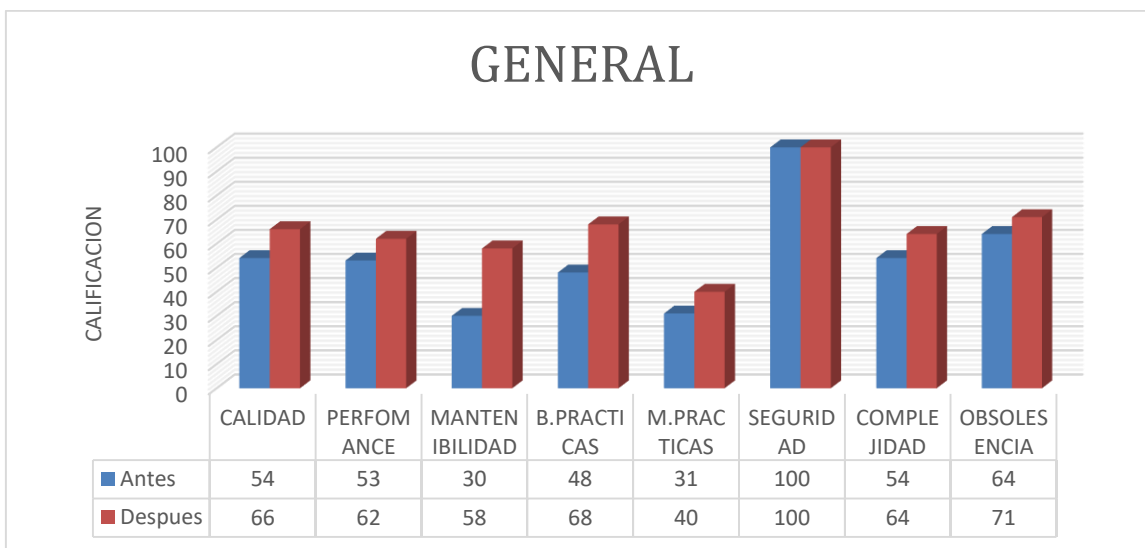
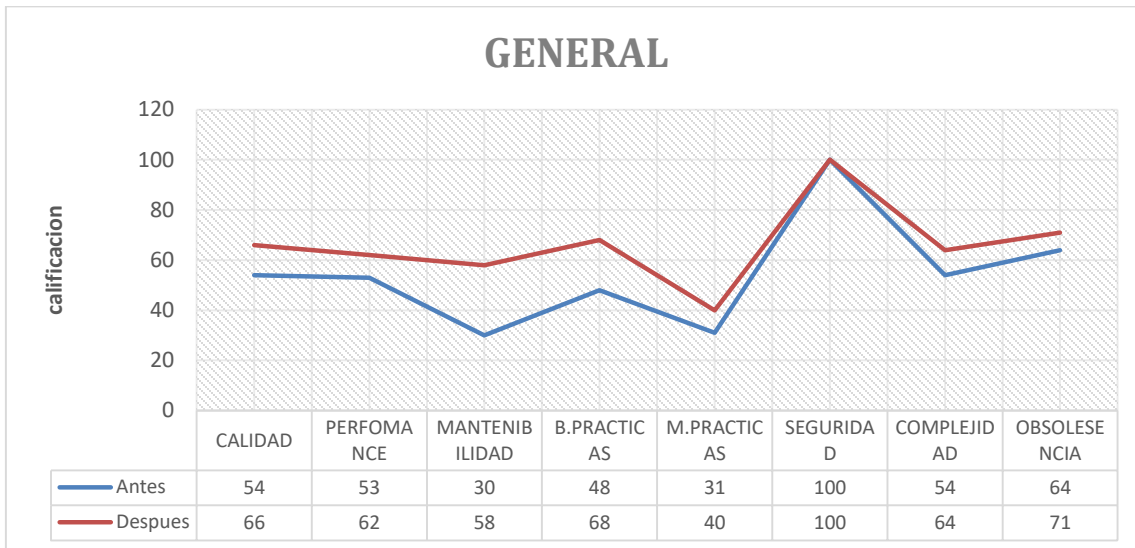


Figura37. Mantenibilidad de Calidad.

Como se puede apreciar en las figura37, existe una tendencia de crecimiento en todos los valores de las variables medidas en los programas analizados, lo que permite concluir que luego de una primera iteración de cambio o mejora en los fuentes de los programas, fruto de las observaciones de las métricas de calidad, se ha mejorado el performance de las aplicaciones, su calidad, las buenas prácticas, se mejoró la adopción de buenas prácticas , en cuanto a seguridad no se encontraron tokens relacionados , se mejoró en la adopción de métricas de mejora en la

complejidad y en obsolescencia no se encontró métricas que relacionaran los programas con esto.

De la misma manera se ve en cada grafico para los programas en particular.

Estos resultados apoyan la hipótesis general de la tesis de que el uso de la herramienta SOURCE400, mejora la calidad de las aplicaciones desarrolladas en lenguaje RPG.

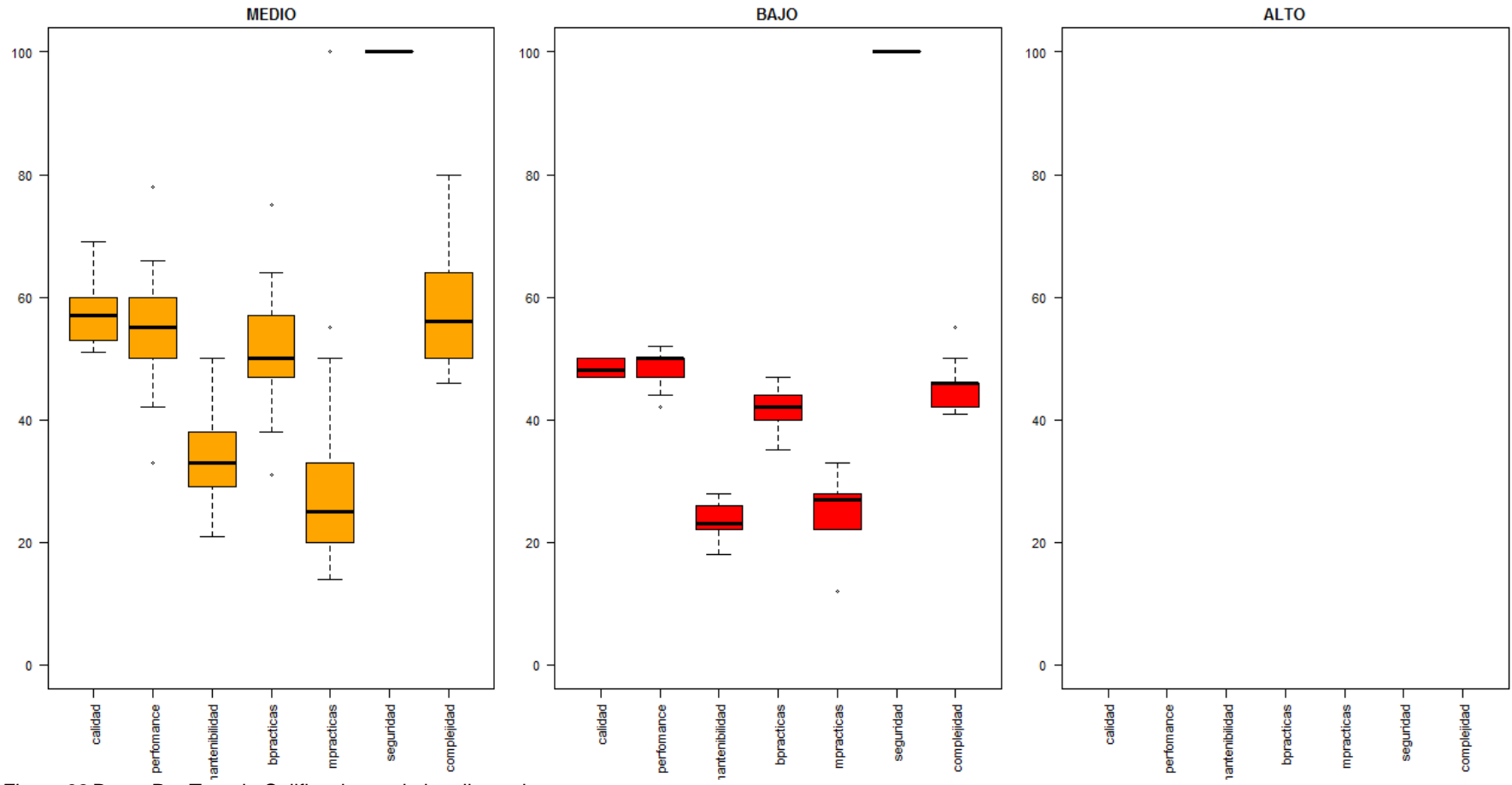


Figura 38. Datos Pre Test de Calificaciones de las dimensiones.

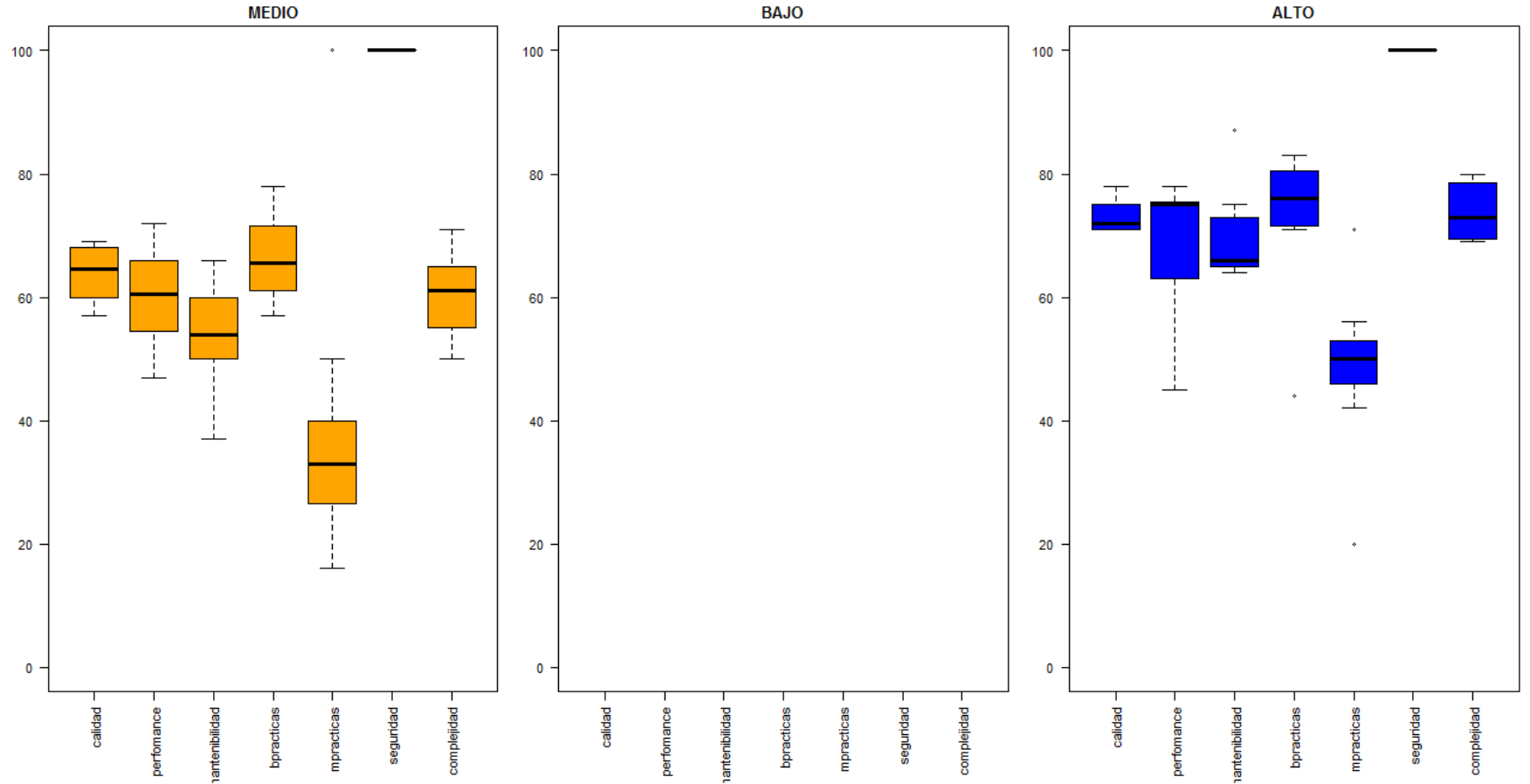


Figura 39. Datos Pos Test de Calificaciones de las dimensiones.

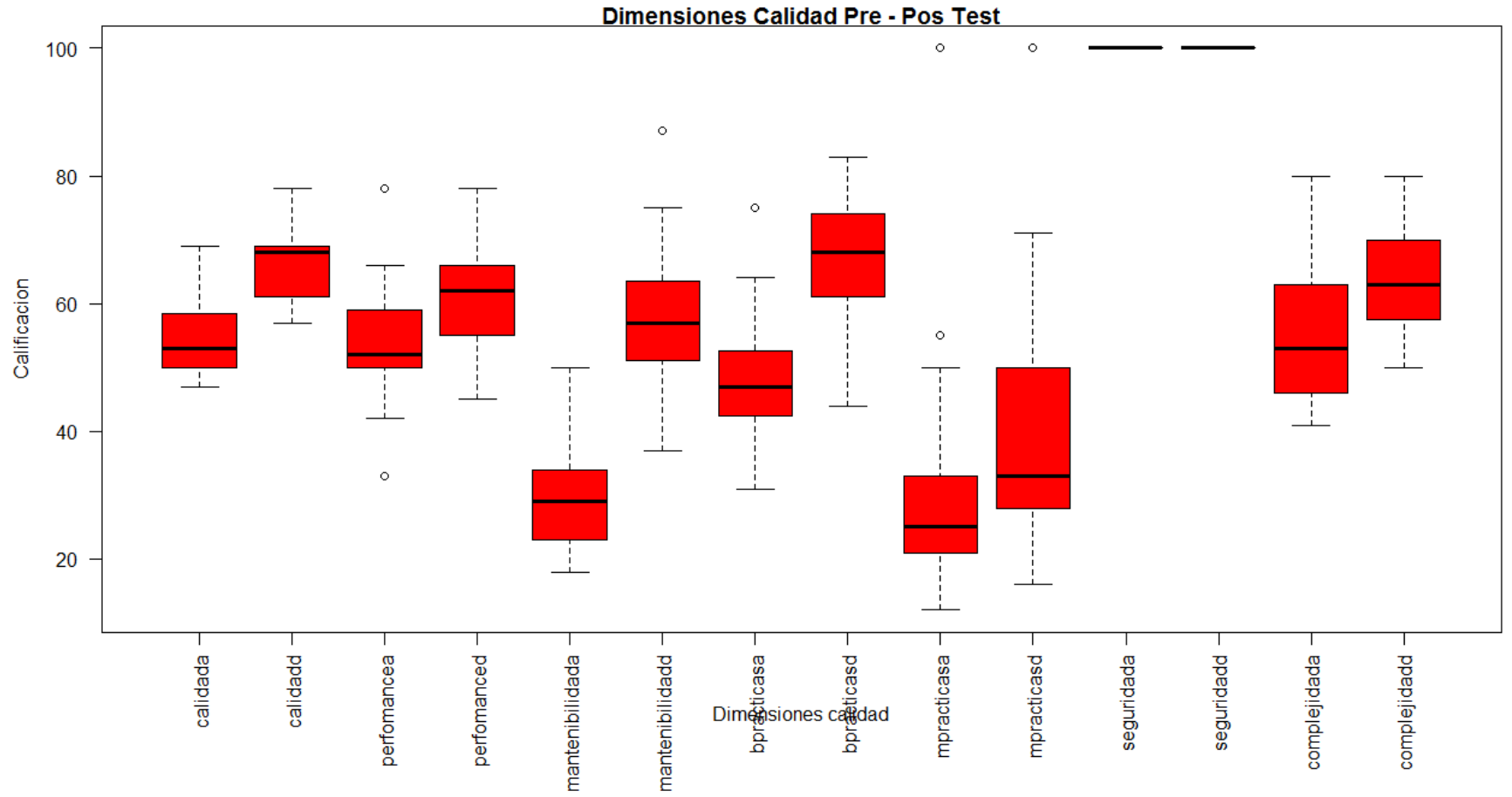


Figura 40.Box plot de calificaciones pre – pos test para las dimensiones.

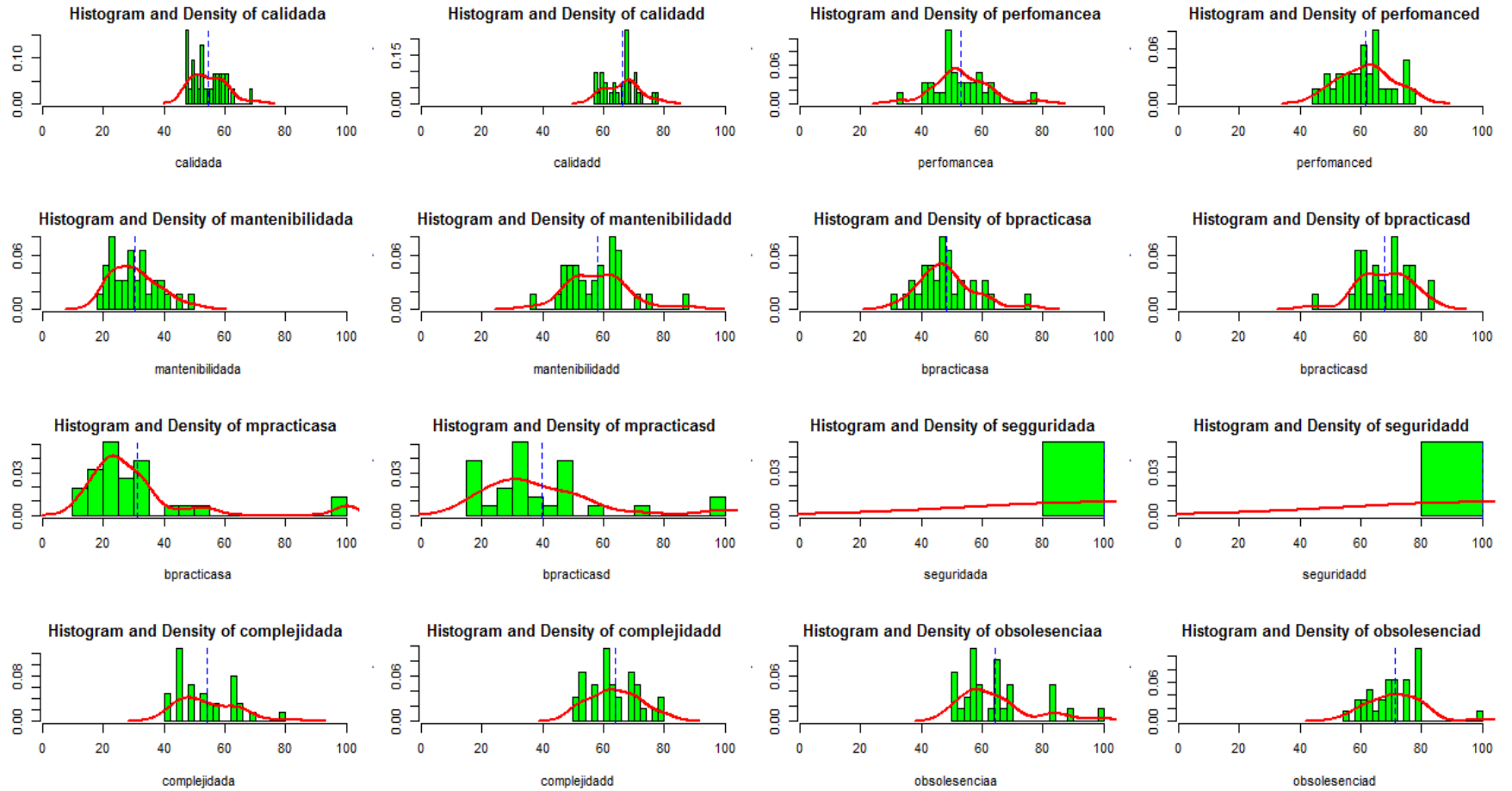


Figura 41. Histogramas y densidad pre y pos test.

El grafico 38 muestra los datos del pre test, se ve datos para las dimensiones con calificaciones bajo y medio por debajo de la calificación 60, en el grafico 39 se observan los datos del pos test donde se nota datos para dimensiones por encima de calificación 60 para calificaciones media y alta, ya no se ven dimensiones con calificaciones bajo, lo que denota mejora en los valores dimensionales.

El Grafico 40 permite ver la comparación entre los resultados del pre y pos test, donde se verifica la mejora de la calidad y de las calificaciones para las dimensiones, dando soporte a la conclusión de que QSOURCE, mejora la calidad del software desarrollado en RPG.

En la figura 41 se muestran los histogramas y densidad pre y pos test. Se puede ver la diferencia en la media para pre y pos test, notándose una mejora.

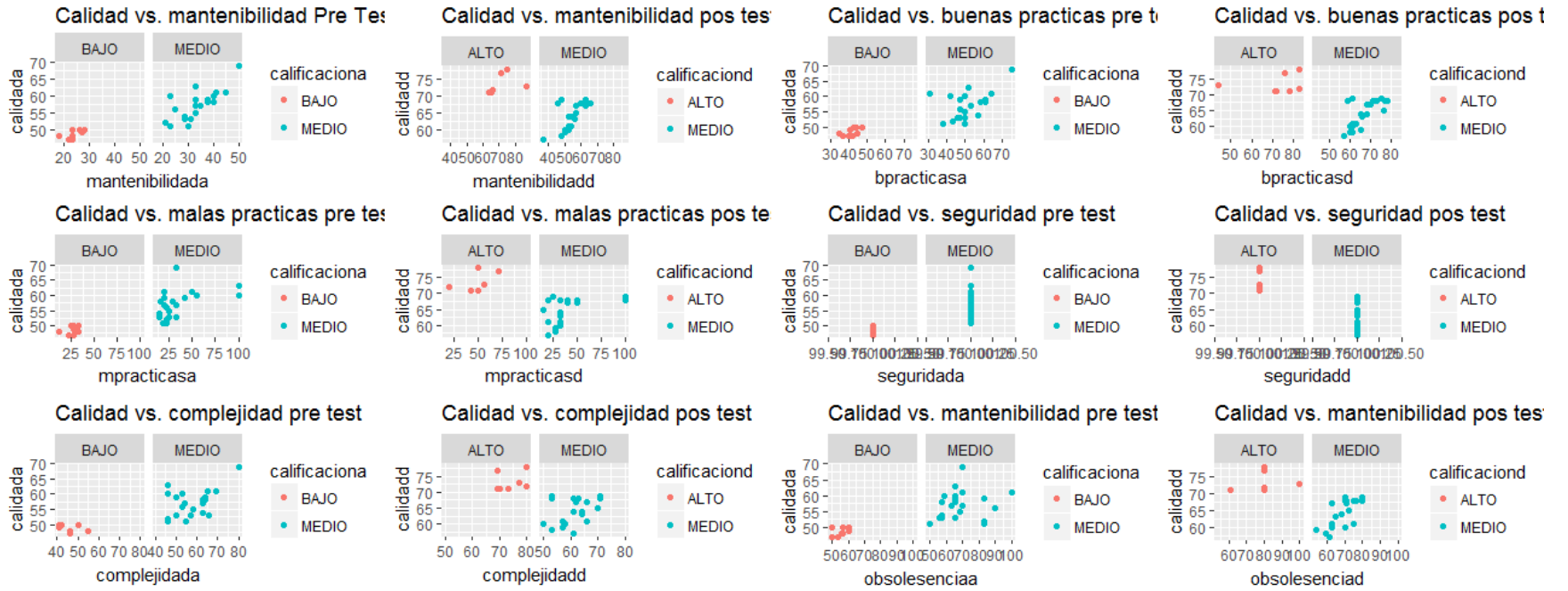


Figura42. ScatterPlots de calidad pre-post test.

En la figura 42 se muestra la dispersión de los datos para la variable calidad contra los diversos indicadores, se puede concluir que con el indicador seguridad no hay relación significativa ya que sus valores son constantes, para los demás indicadores se ve una tendencia para las calificaciones alta y medias.

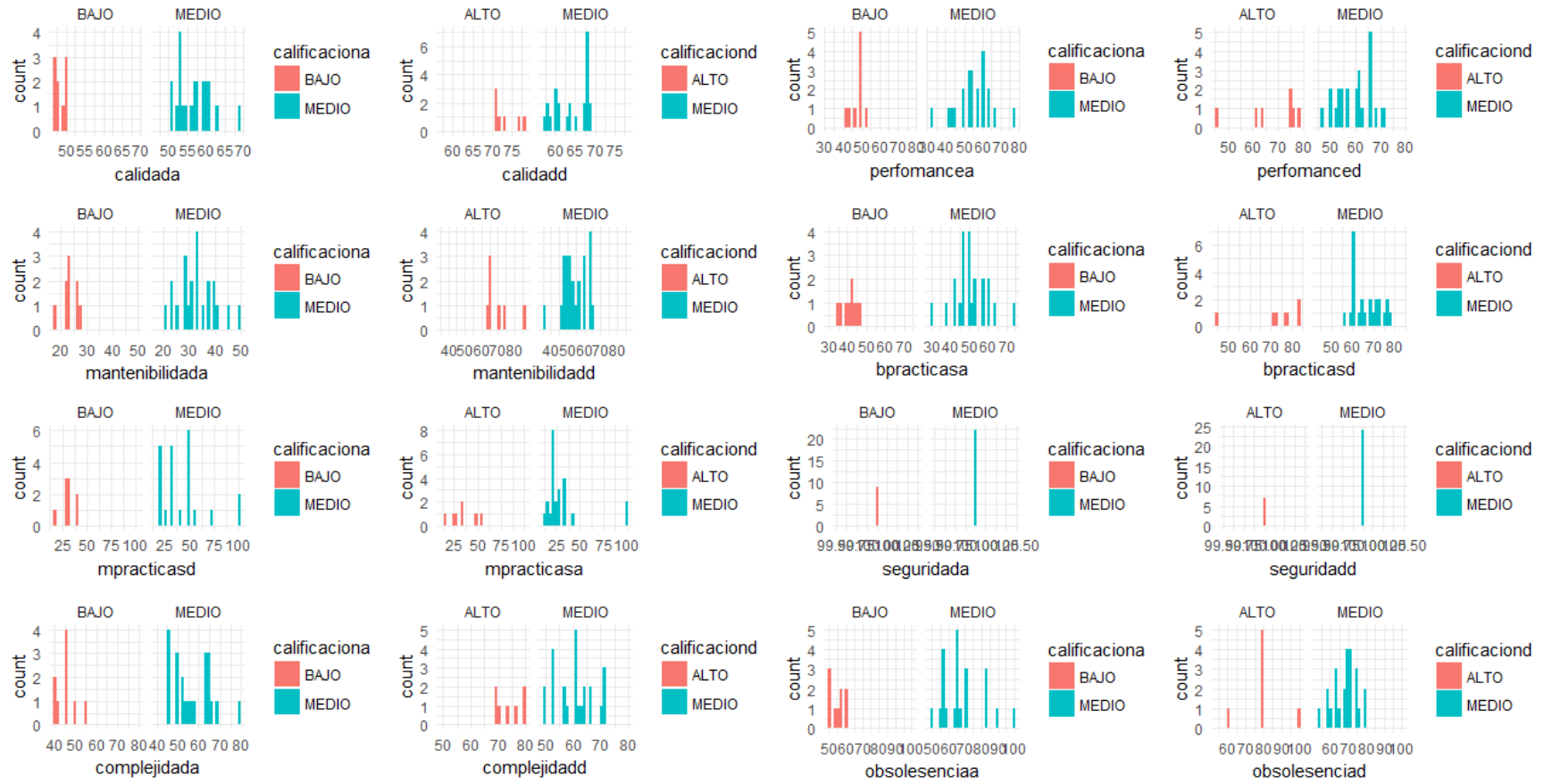


Figura43. Histogramas de calidad vs indicadores pre-post test.

En la figura 43 se muestra los histogramas de los datos para la variable calidad contra los diversos indicadores, se puede concluir que con el indicador seguridad no hay relación significativa ya que sus valores son constantes, para los demás indicadores se ve una tendencia para las calificaciones alta y medias.

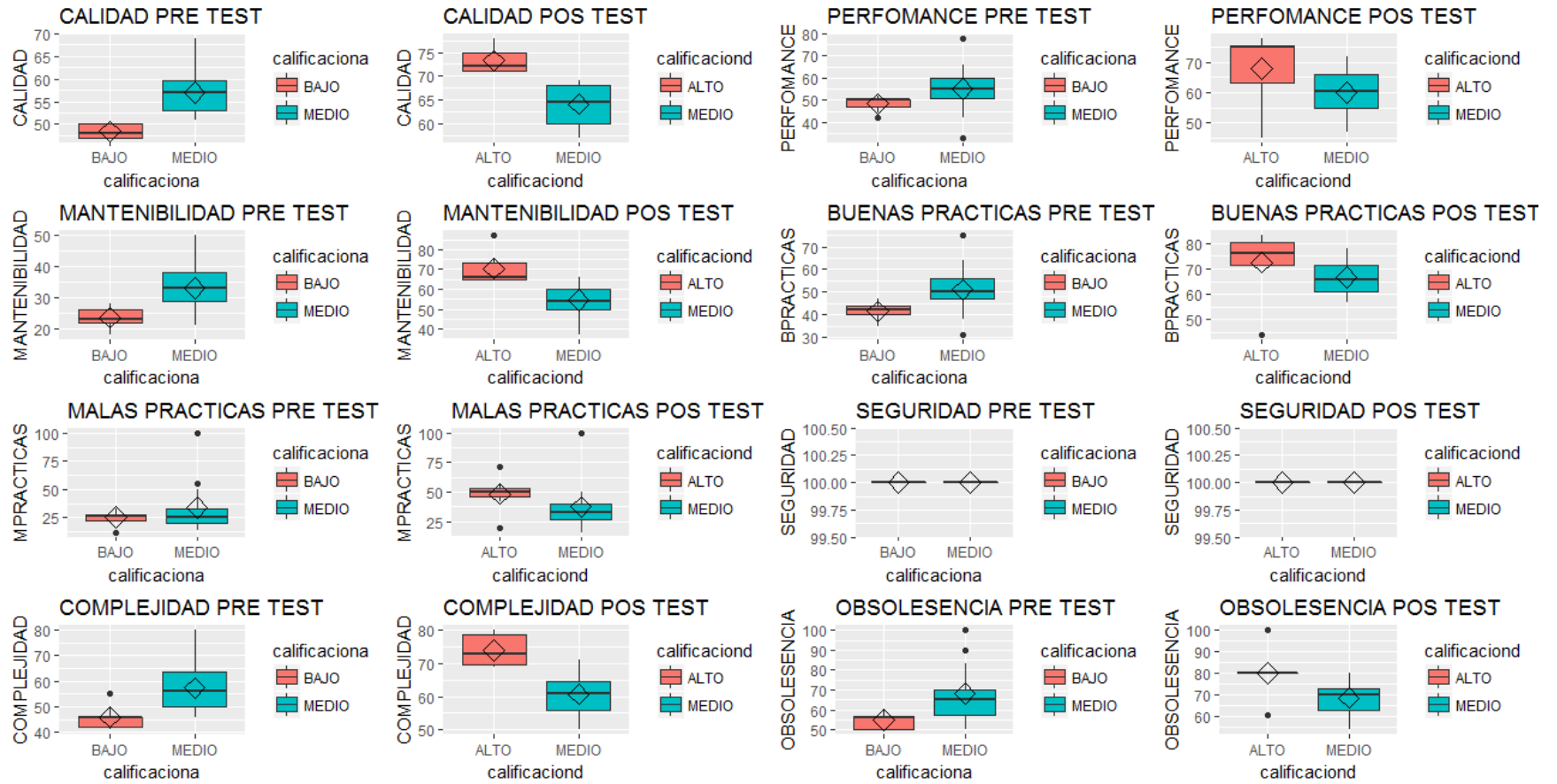


Figura44. Box plot general pre-post test.

En la figura 44 se muestra los boxplots de los datos para la variable calidad y los diversos indicadores, se puede concluir que se ha mejorado ya que los valores con calificación bajo no existen y se ve valores mayores para calificaciones alto y medio.

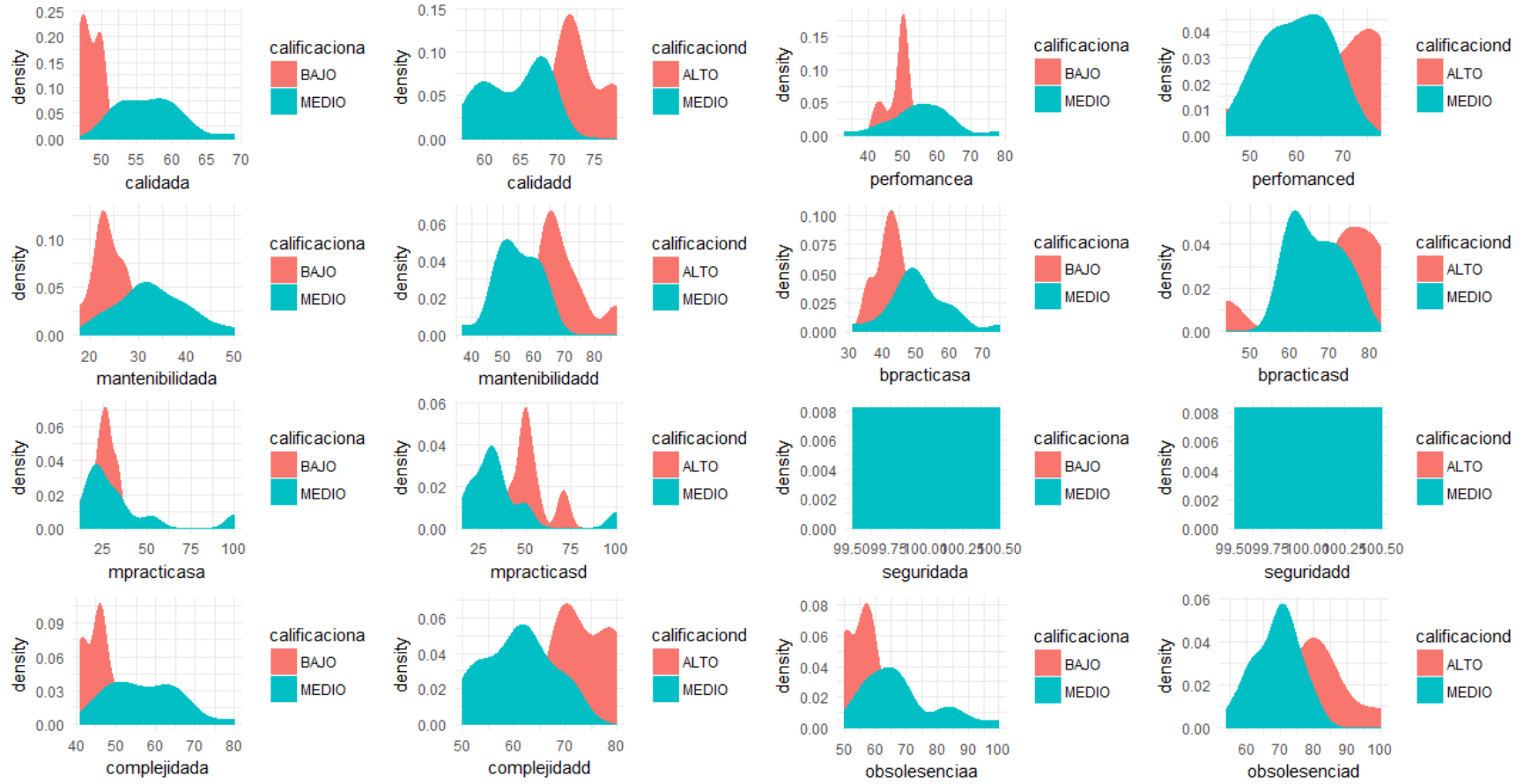


Figura45. Density plots pre-post test.

En la figura 45 se muestra los gráficos de probabilidad de densidad de los datos para la variable calidad contra los diversos indicadores, se puede concluir que con el indicador seguridad no hay relación significativa ya que sus valores son constantes, para los demás indicadores se ve una tendencia para las calificaciones alta y medias.

La distribución de los datos para calificación baja es pequeña en comparación con la calificación alta que tiene probabilidad baja, pero densidad alta, en cuanto a la calificación media, su probabilidad es media y su densidad es alta.

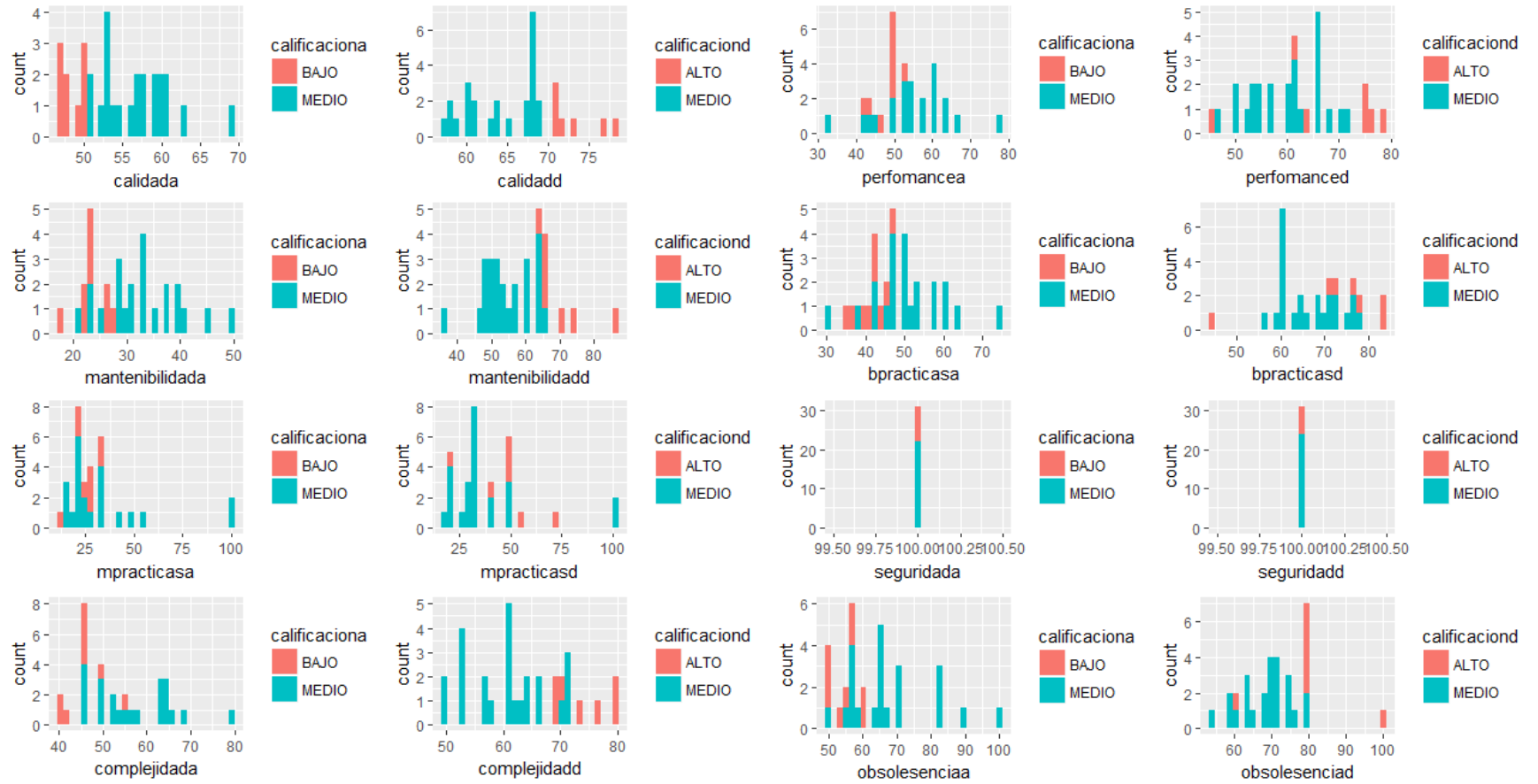


Figura46. Histogramas de calidad e indicadores pre-post test.

En la figura 46 se muestra los histogramas de los datos para la variable calidad contra los diversos indicadores, se puede concluir que con el indicador seguridad no hay relación significativa ya que sus valores son constantes, para los demás indicadores se ve una tendencia para las calificaciones alta y medias.

Las frecuencias más altas se dan para calificaciones altas y medias, y las frecuencias más bajas son para programas con calificación bajo.

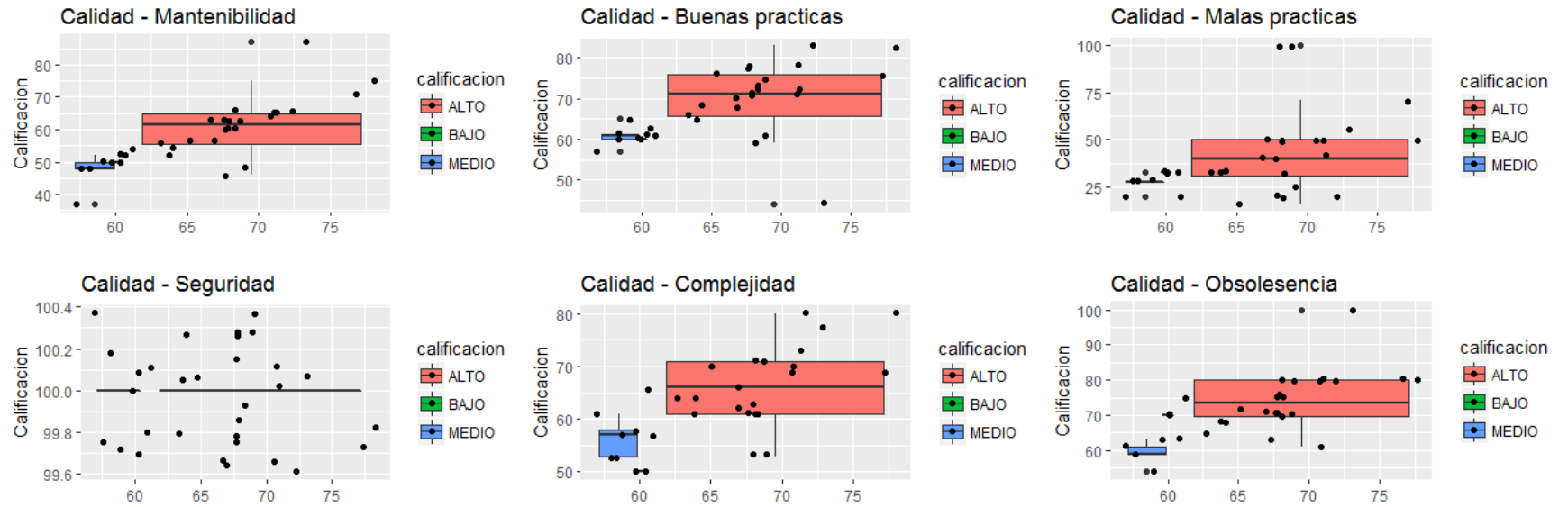
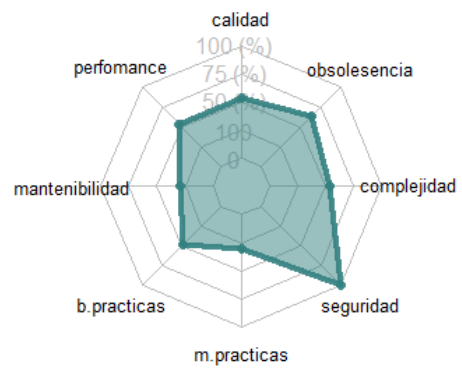


Figura47.Boxplots calidad vs indicadores post test.

En la figura 47 se muestra los box plots de los datos para la variable calidad contra los diversos indicadores, se puede concluir que con el indicador seguridad no hay relación significativa ya que sus valores son constantes, para los demás indicadores se ve una tendencia para las calificaciones alta y medias.

La distribución del conjunto de datos tiende a ser simétrica, como se ve el 50% de la población está en los límites de la caja, buenas prácticas, complejidad y obsolescencia tienen los mayores valores.

Resultados Generales Pre Test



Resultados Generales Pos Test

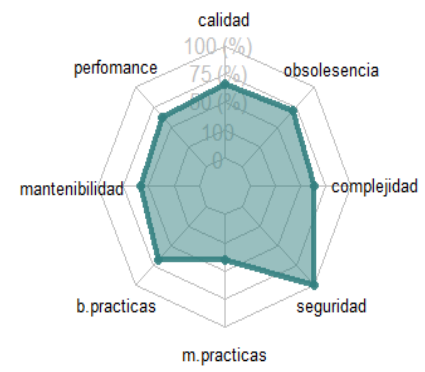


Figura48. Graficas polares de resultados generales.

Como se puede apreciar en la figura 48, existe una tendencia de crecimiento en todos los valores de las variables medidas en los programas analizados, lo que permite concluir que luego de una primera iteración de cambio o mejora en los fuentes de los programas, fruto de las observaciones de las métricas de calidad, se ha mejorado el performance de las aplicaciones, su calidad, las buenas prácticas, se mejoró la adopción de buenas prácticas, en cuanto a seguridad no se encontraron tokens relacionados, se mejoró en la adopción de métricas de mejora en la complejidad y en obsolescencia no se encontró métricas que relacionaran los programas con esto.

De la misma manera se ve en cada gráfico para los programas en particular.

Estos resultados apoyan la hipótesis general de la tesis de que el uso de la herramienta SOURCE400, mejora la calidad de las aplicaciones desarrolladas en lenguaje RPG.

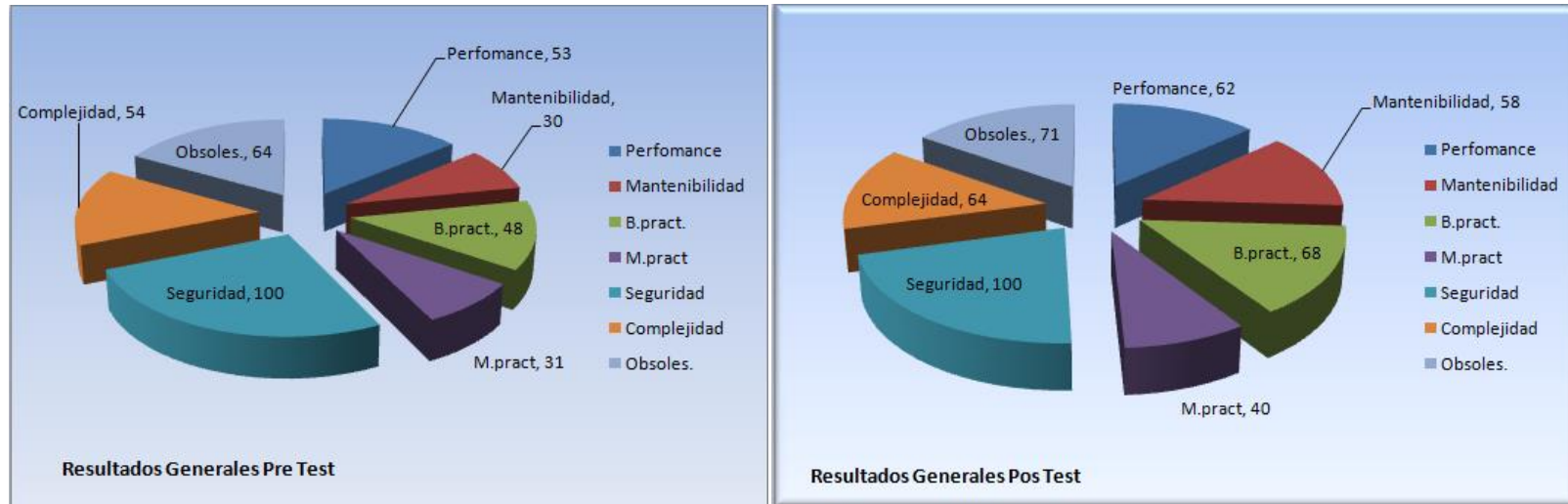


Figura49. Resultados Generales, gráficos de torta.

3.11 Resultados generales por variable

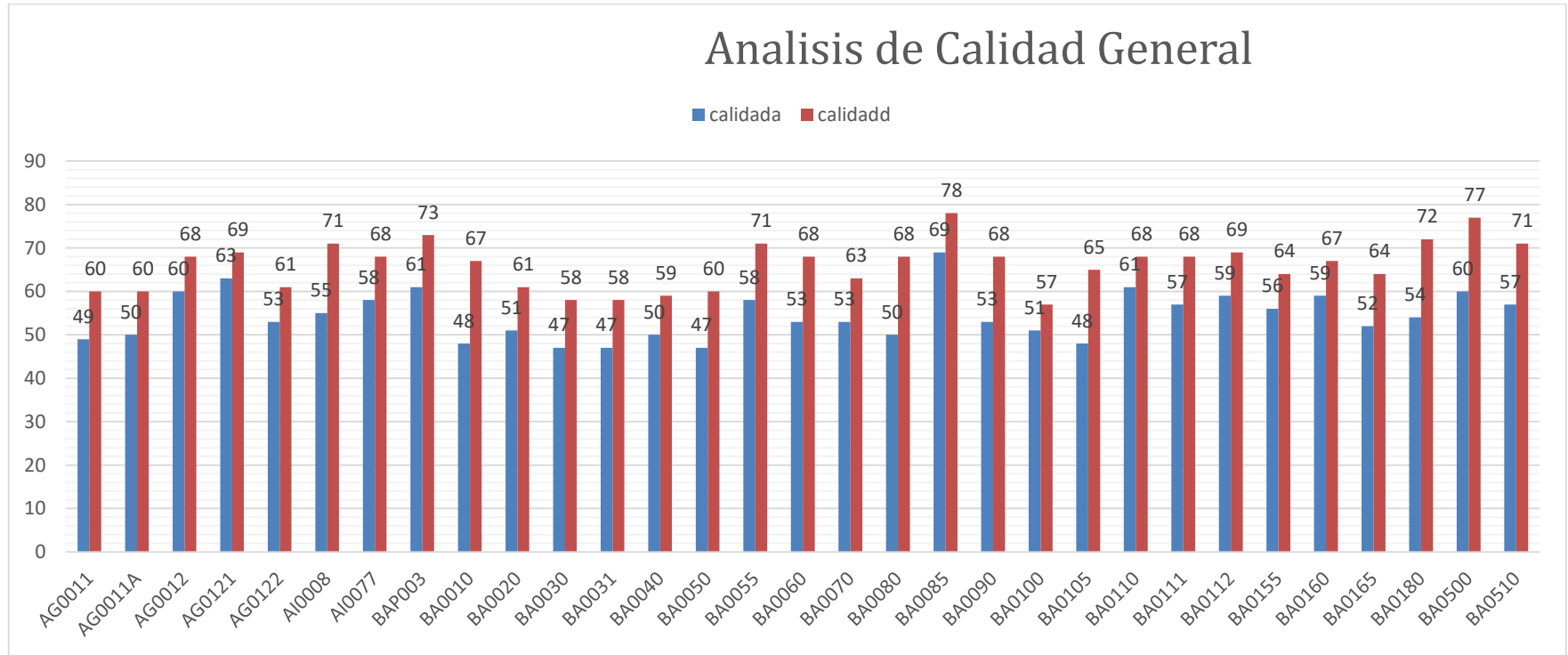


Figura50. Análisis de calidad general.

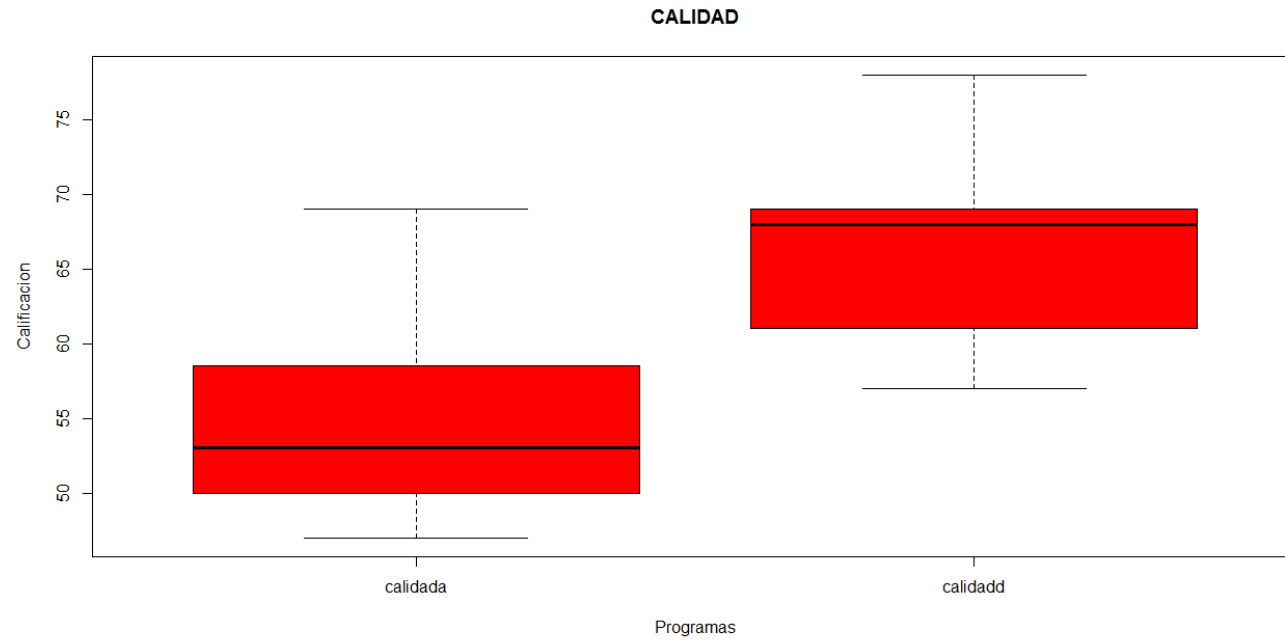


Figura51. Análisis de calidad general.

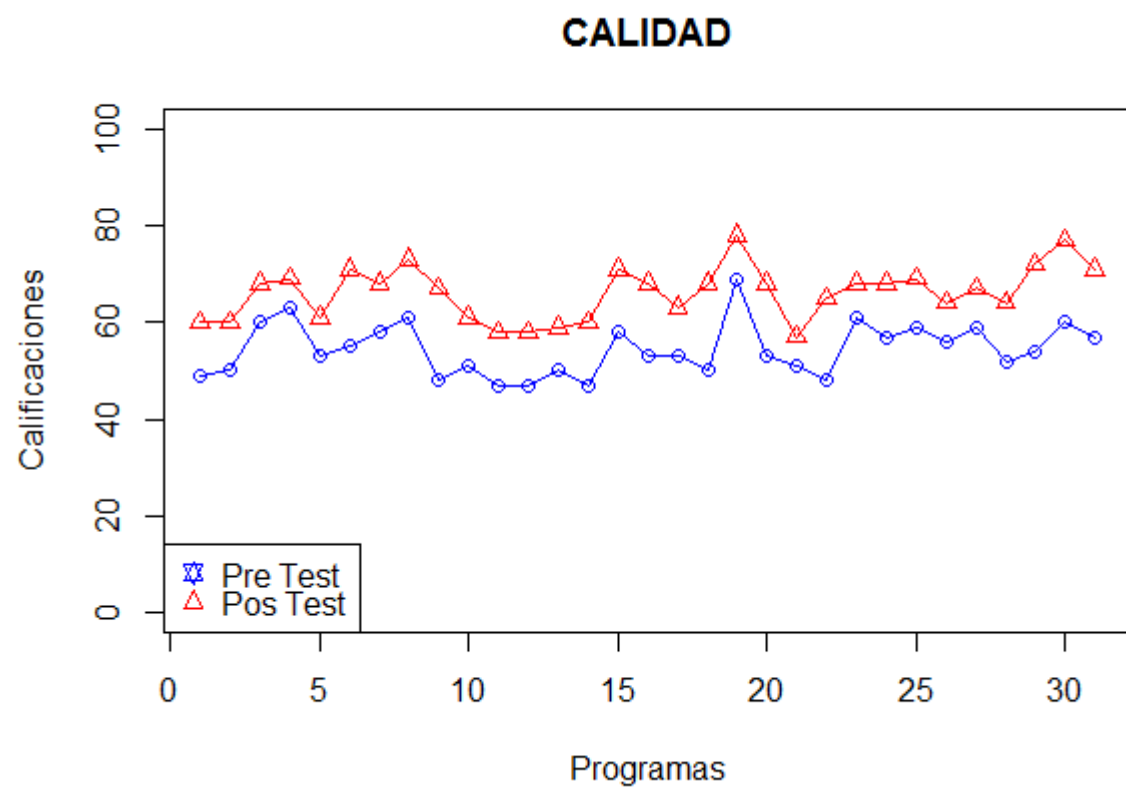


Figura 52. Análisis de calidad general.

Como se puede apreciar en la gráfica 50. existe una tendencia de crecimiento en los valores de la variable calidad medida en los programas analizados, lo que permite concluir que luego de una primera iteración la calidad cambio o mejora en los fuentes de los programas, fruto de las observaciones de las métricas de calidad se ha mejorado la gestión de calidad de las aplicaciones estos resultados apoyan la hipótesis general de la tesis de que el uso de la herramienta SOURCE400, mejora la gestión de calidad de las aplicaciones desarrolladas en lenguaje RPG.El grafico 51 muestra que la mediana para la variable calidad es mayor en el pos test, lo que muestra un promedio mayor de programas con calificación (70%) por encima de 60 del pre test. El grafico 52 muestra picos o conglomerados más altos en las barras 8,20, 30pres test y post test que son los valores más comunes, la dispersión de los datos es desde 0 hasta 100. El histograma tiene una línea de distribución ajustada, sigue las alturas de las barras, los datos se ajustan a la distribución.

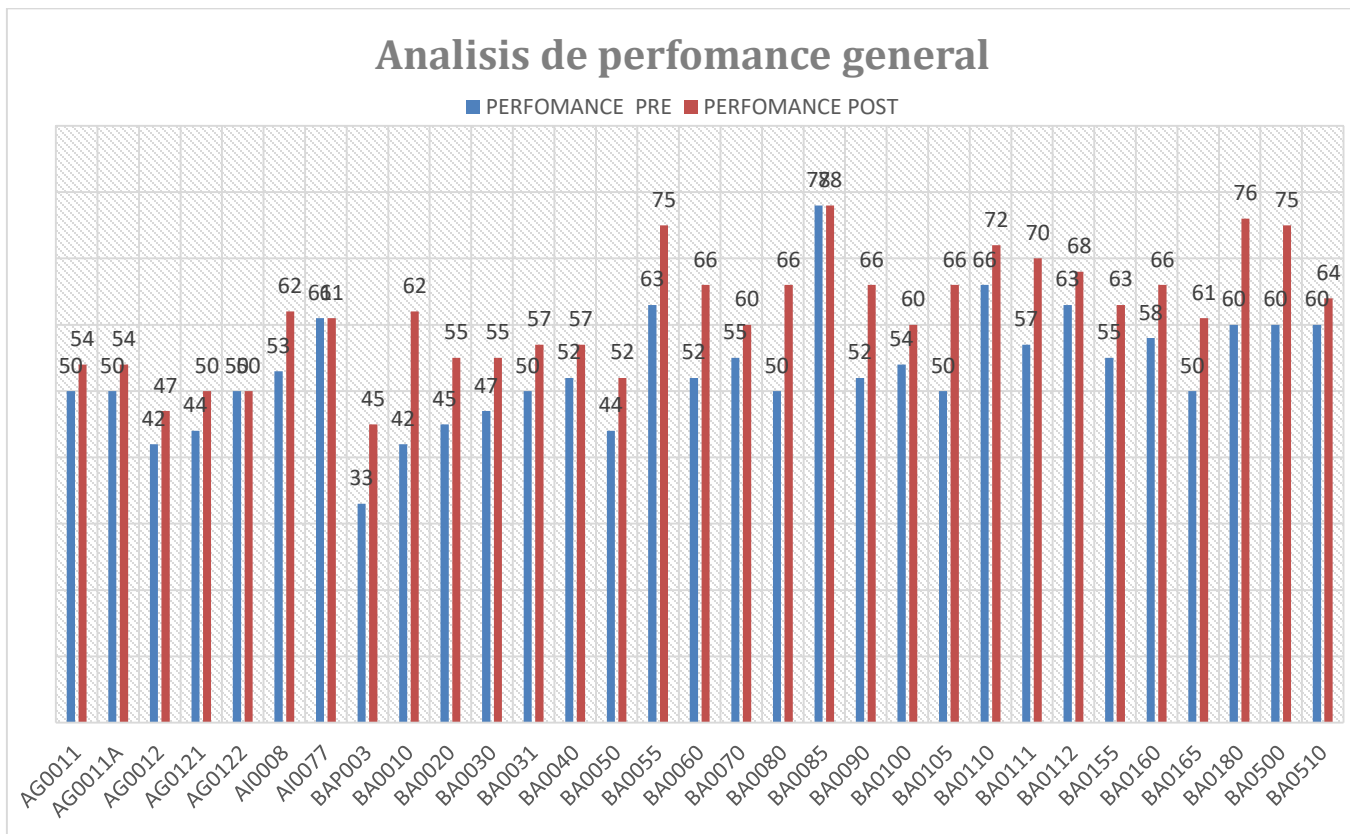


Figura53. Análisis de performance general.

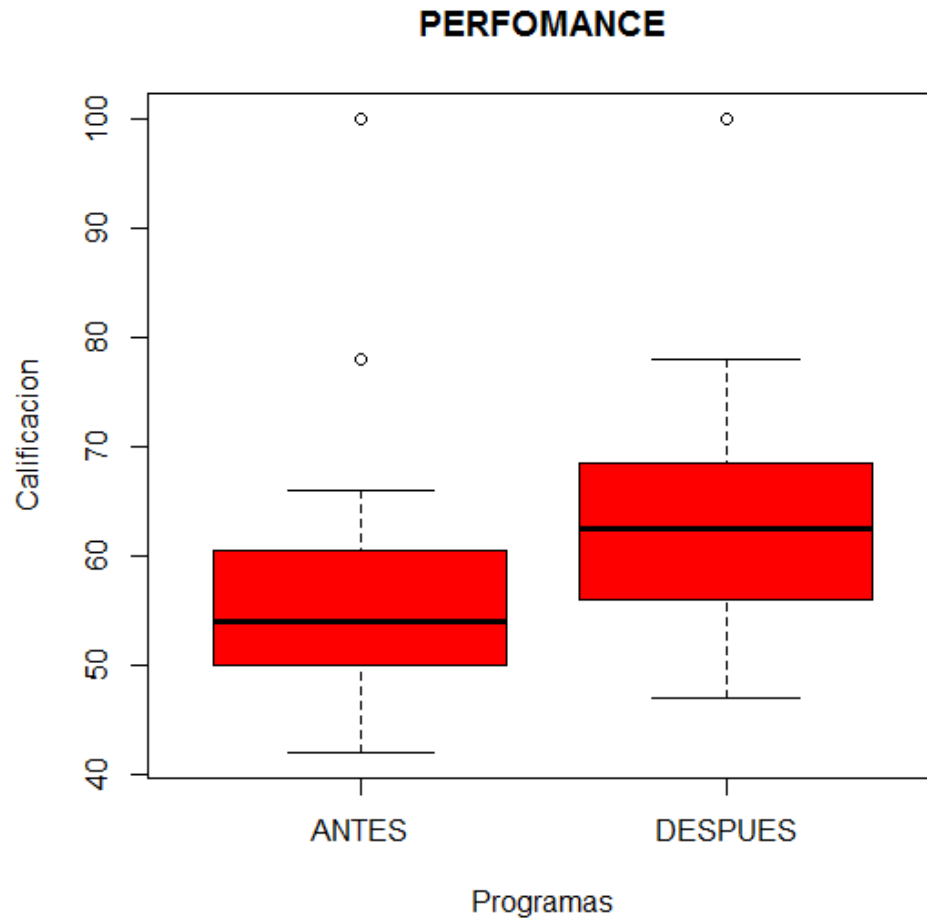


Figura54. Análisis de performance general.

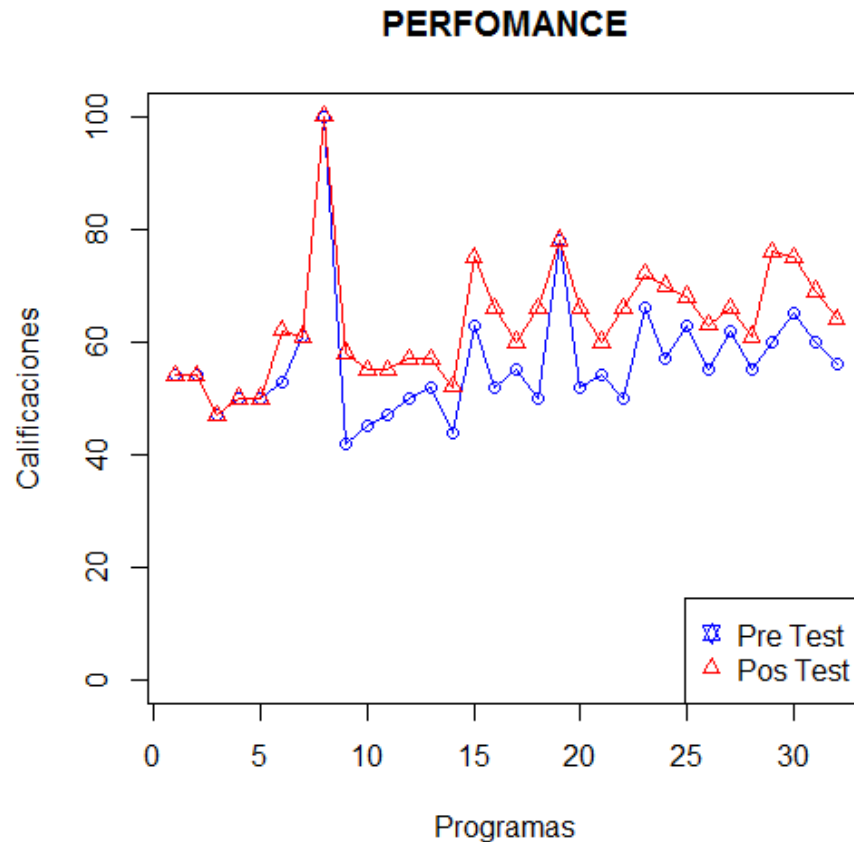


Figura55. Grafico de puntos pre y pos tes para variable performance.

Como se puede apreciar en la gráfica 53. existe una tendencia de crecimiento en los valores de la variable PERFOMANCE medida en los programas analizados, lo que permite concluir que luego de una primera iteración d PERFOMANCE e cambio o mejora en los fuentes de los programas, fruto de las observaciones de las métricas de PERFOMANCE se ha mejorado la gestión de PERFOMANCE de las aplicaciones estos resultados apoyan la hipótesis general de la tesis de que el uso de la herramienta SOURCE400, mejora la gestión de PERFOMANCE de las aplicaciones desarrolladas en lenguaje RPG.El grafico 54 muestra que la mediana para la variable Perfomance es mayor en el pos test, lo que muestra un promedio mayor de programas con calificación (75%) por encima de 60 contra 53 del pre test. El grafico 55 muestra picos o conglomerados más altos en las barras 50-55 pre test, 65-70 post test que son los valores más comunes, la dispersión de los datos es desde40 hasta 100. El histograma

tiene una línea de distribución ajustada, sigue las alturas de las barras, los datos se ajustan a la distribución.

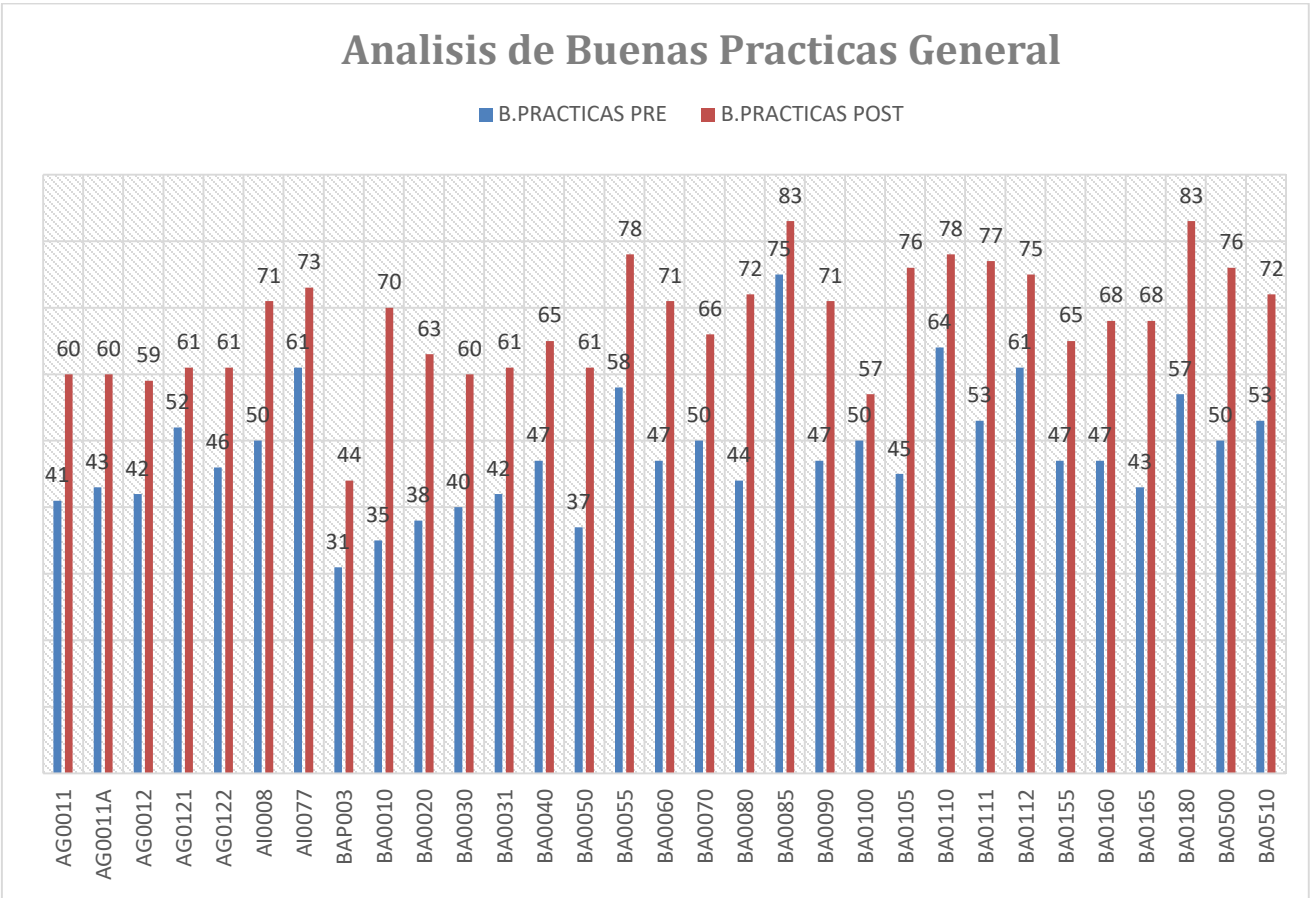


Figura56. Análisis de Buenas prácticas general.

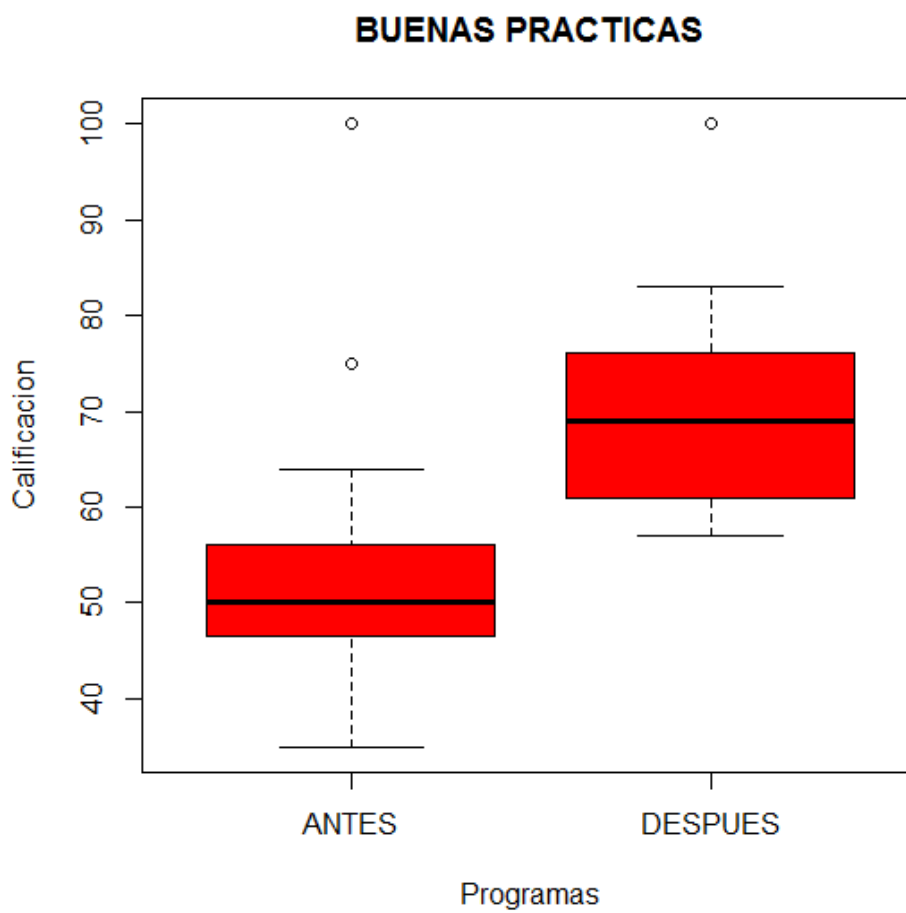


Figura57. Análisis de Buenas prácticas general.

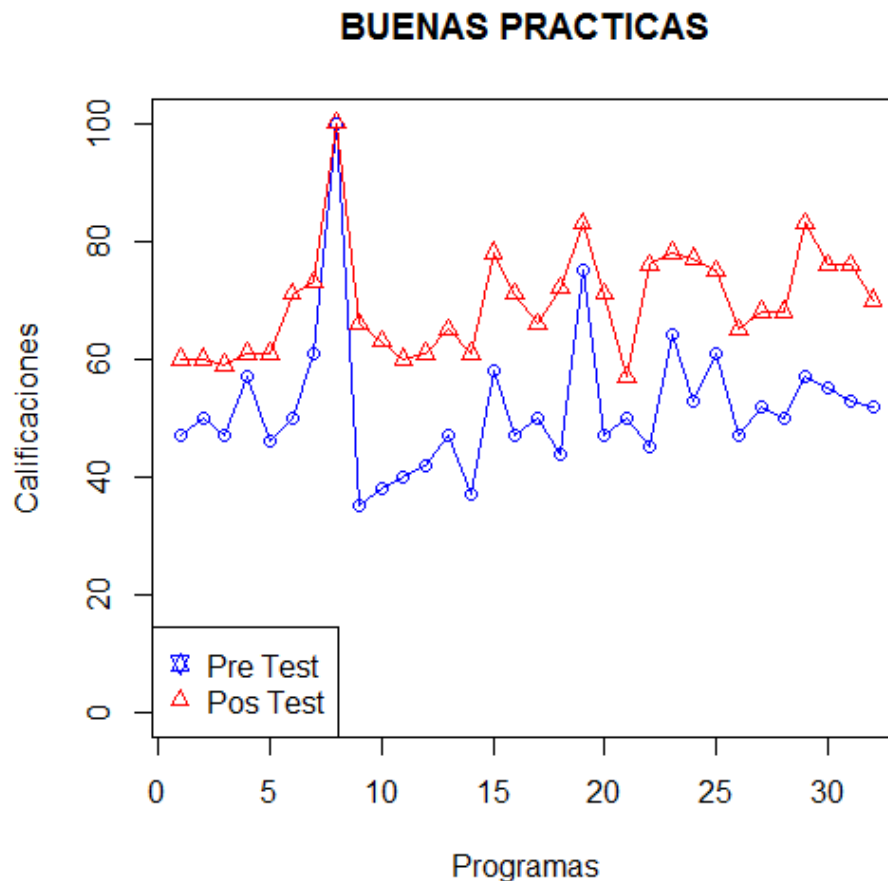


Figura58. Grafico de puntos pre y pos tes para variable buenas prácticas.

Como se puede apreciar en la gráfica56 existe una tendencia de crecimiento en los valores de la variable BUENAS PRACTICAS medida en los programas analizados, lo que permite concluir que luego de una primera iteración de cambio o mejora en los fuentes de los programas, fruto de las observaciones de las métricas de BUENAS PRACTICAS, se ha mejorado la gestión de BUENAS PRACTICAS de las aplicaciones estos resultados apoyan la hipótesis general de la tesis de que el uso de la herramienta SOURCE400, mejora la gestión de BUENAS PRACTICAS de las aplicaciones desarrolladas en lenguaje RPG.El grafico 57 muestra que la mediana para la variable buenas prácticas es mayor en el pos test, lo que muestra un promedio mayor de programas con calificación (75%) por encima de 70 contra valores inferiores a 50 del pre test. El grafico 58 muestra picos o conglomerados más altos en las barras 45-50 pre test, 60-65 post test que son los valores más comunes, la dispersión de los

datos es desde 40 hasta 100. El histograma tiene una línea de distribución ajustada, sigue las alturas de las barras, los datos se ajustan a la distribución.

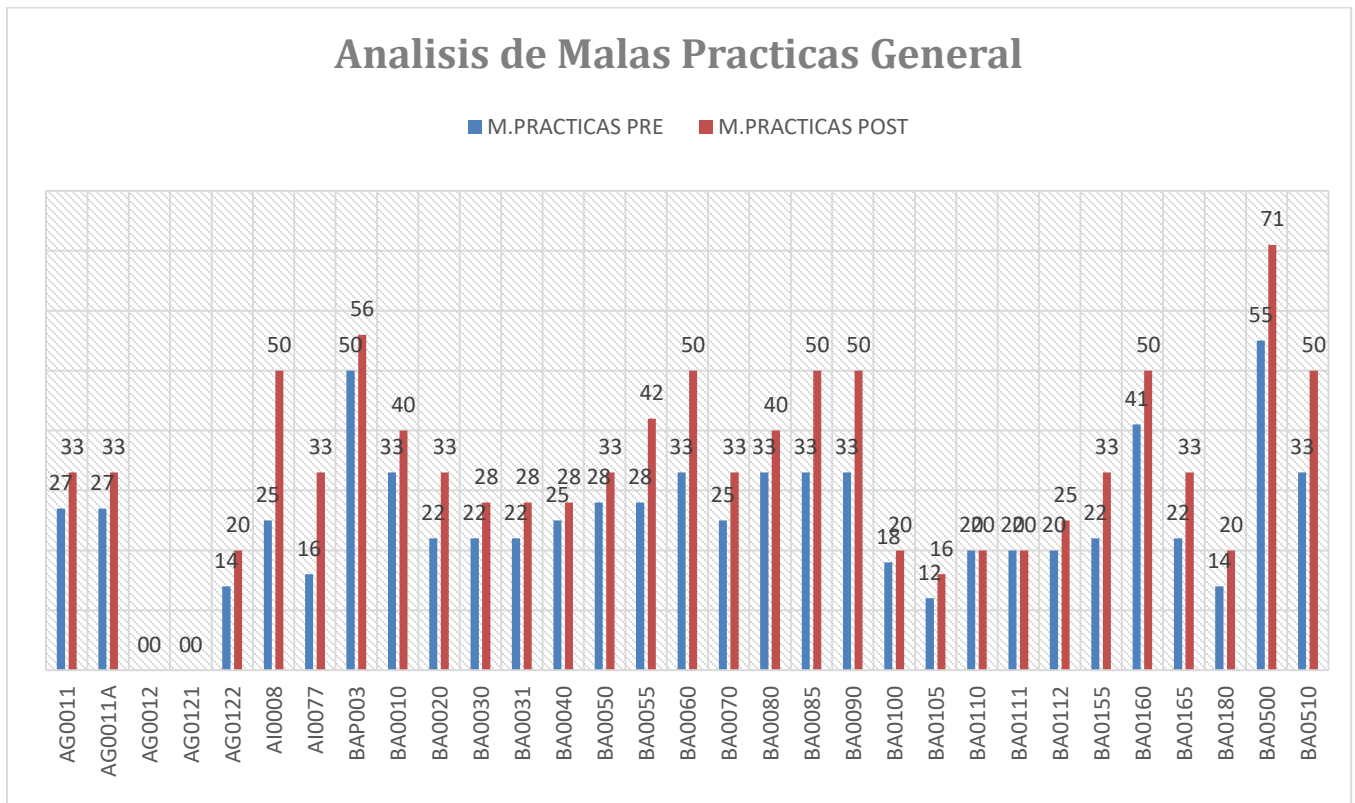


Figura59. Análisis de malas prácticas general.

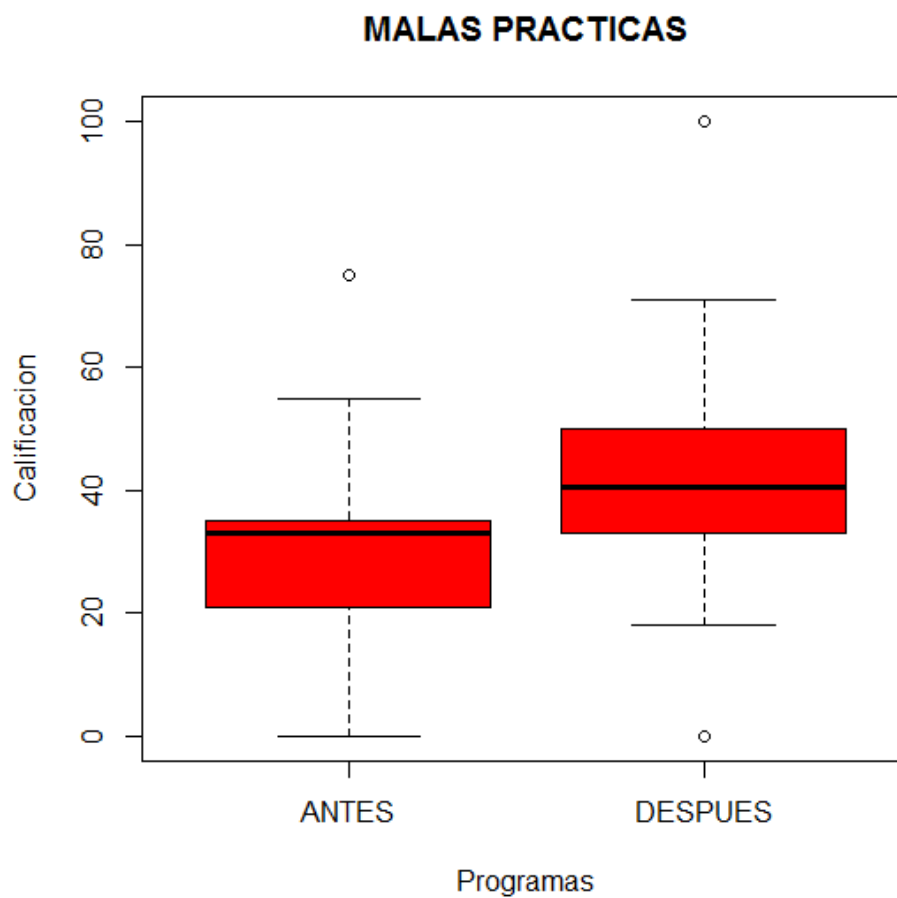


Figura60. Análisis de malas prácticas general.

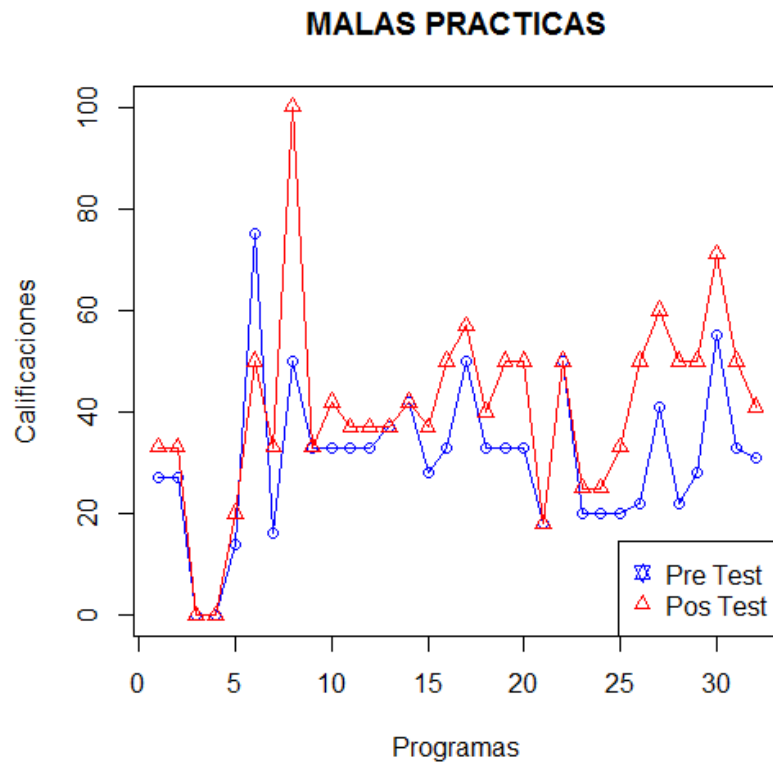


Figura61. Grafico de puntos pre y pos tes para variable malas prácticas.

Como se puede apreciar en la gráfica59, existe una tendencia de crecimiento en los valores de la variable MALAS PRACTICAS medida en los programas analizados, lo que permite concluir que luego de una primera iteración de cambio o mejora en los fuentes de los programas, fruto de las observaciones de las métricas de MALAS PRACTICAS, se ha mejorado la gestión de MALAS PRACTICAS de las aplicaciones estos resultados apoyan la hipótesis general de la tesis de que el uso de la herramienta SOURCE400, mejora la gestión de MALAS PRACTICAS de las aplicaciones desarrolladas en lenguaje RPG.El grafico 60 muestra que la mediana para la variable malas prácticas es mayor en el pos test, lo que muestra un promedio mayor de programas con calificación (75%) por encima de 40 contra valores inferiores a 40 del pre test. El grafico 61 muestra picos o conglomerados más altos en las barras 35 pre test, 45 post test que son los valores más comunes, la dispersión de los datos es desde 0 hasta 100. El histograma tiene una línea de distribución ajustada, sigue las alturas de las barras, los datos se ajustan a la distribución.

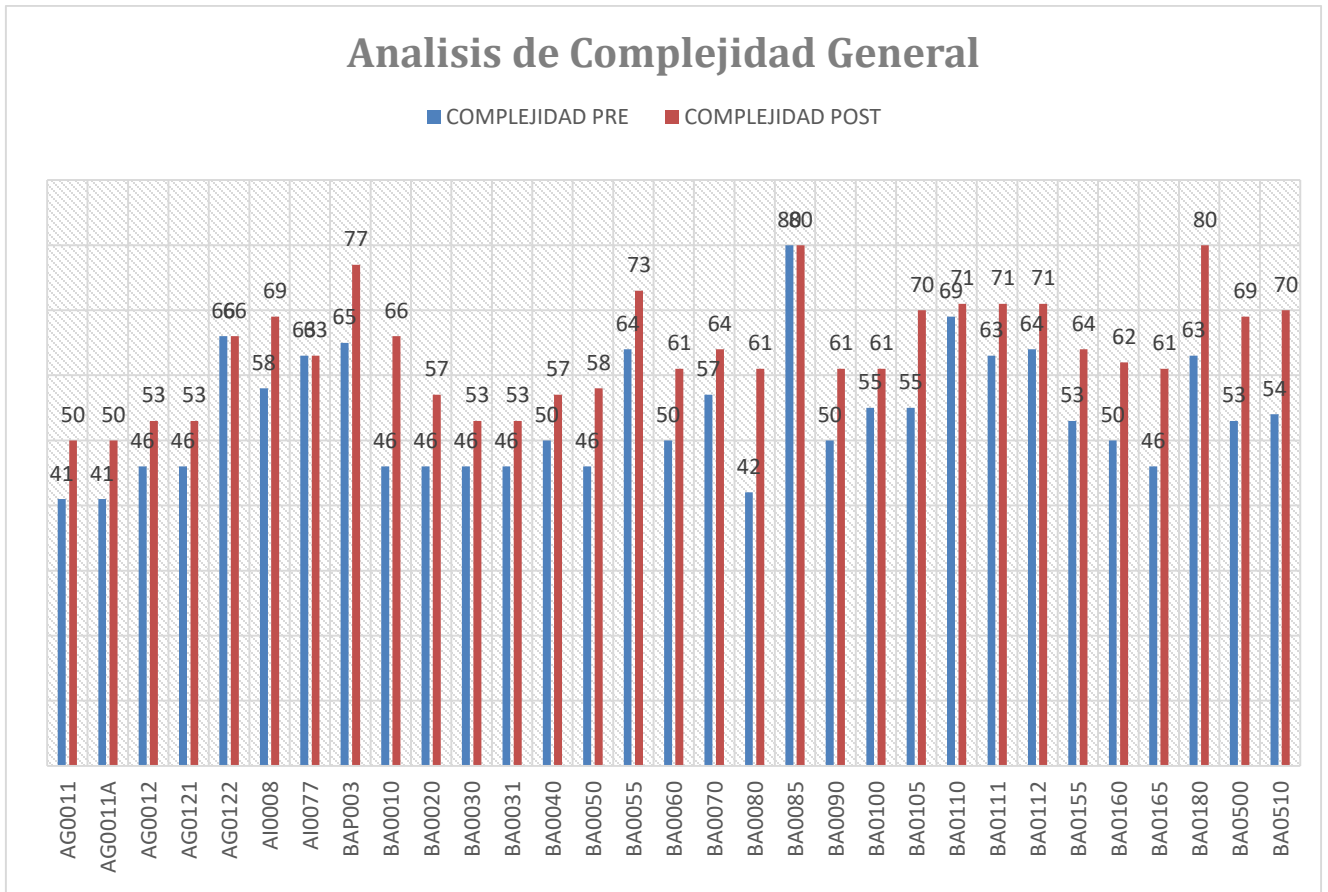


Figura62. Análisis de complejidad general.

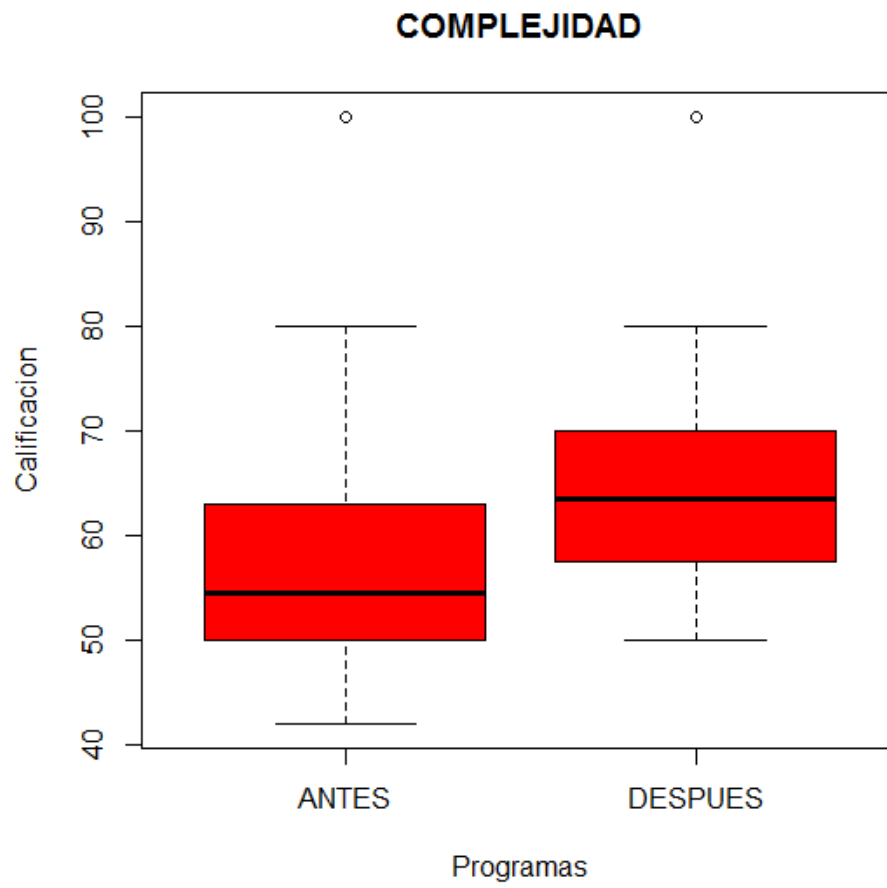


Figura63. Análisis de complejidad general.

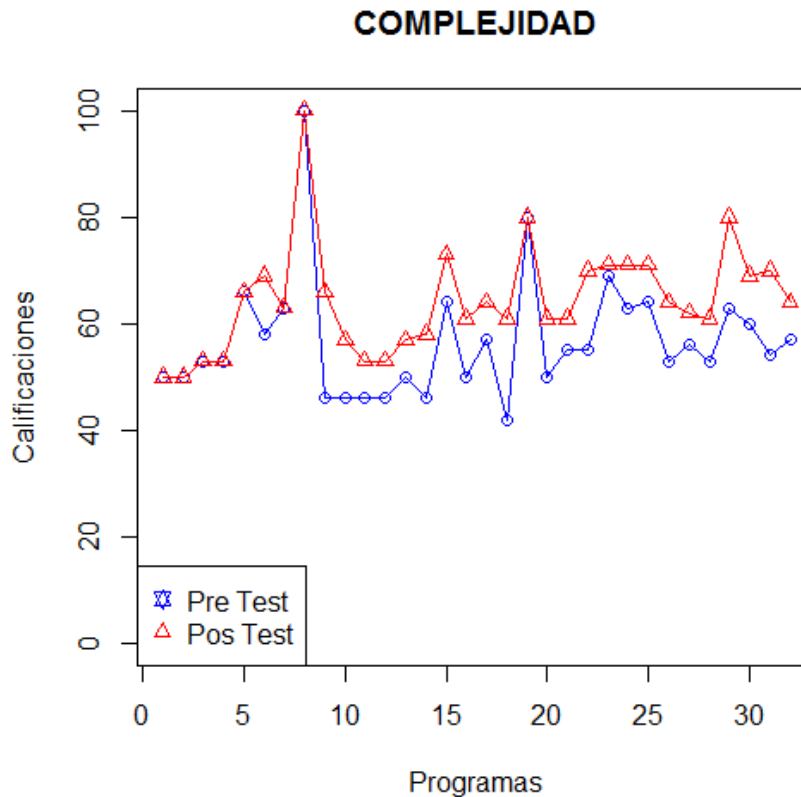


Figura64. Grafico de puntos pre y pos tes para variable complejidad.

Como se puede apreciar en la gráfica62, existe una tendencia de crecimiento en los valores de la variable COMPLEJIDAD medida en los programas analizados, lo que permite concluir que luego de una primera iteración de cambio o mejora en los fuentes de los programas, fruto de las observaciones de las métricas de COMPLEJIDAD, se ha mejorado la COMPLEJIDAD de las aplicaciones estos resultados apoyan la hipótesis general de la tesis de que el uso de la herramienta SOURCE400, mejora la COMPLEJIDAD de las aplicaciones desarrolladas en lenguaje RPG.El grafico 63 muestra que la mediana para la variable complejidad es mayor en el pos test, lo que muestra un promedio mayor de programas con calificación (75%) por encima de 62 contra valores inferiores a 55 del pre test. El grafico 64 muestra picos o conglomerados más altos en las barras 45-50 pre test, 60-65 post test que son los valores más comunes, la dispersión de los datos es desde 40 hasta 100. El histograma

tiene una línea de distribución ajustada, sigue las alturas de las barras, los datos se ajustan a la distribución.

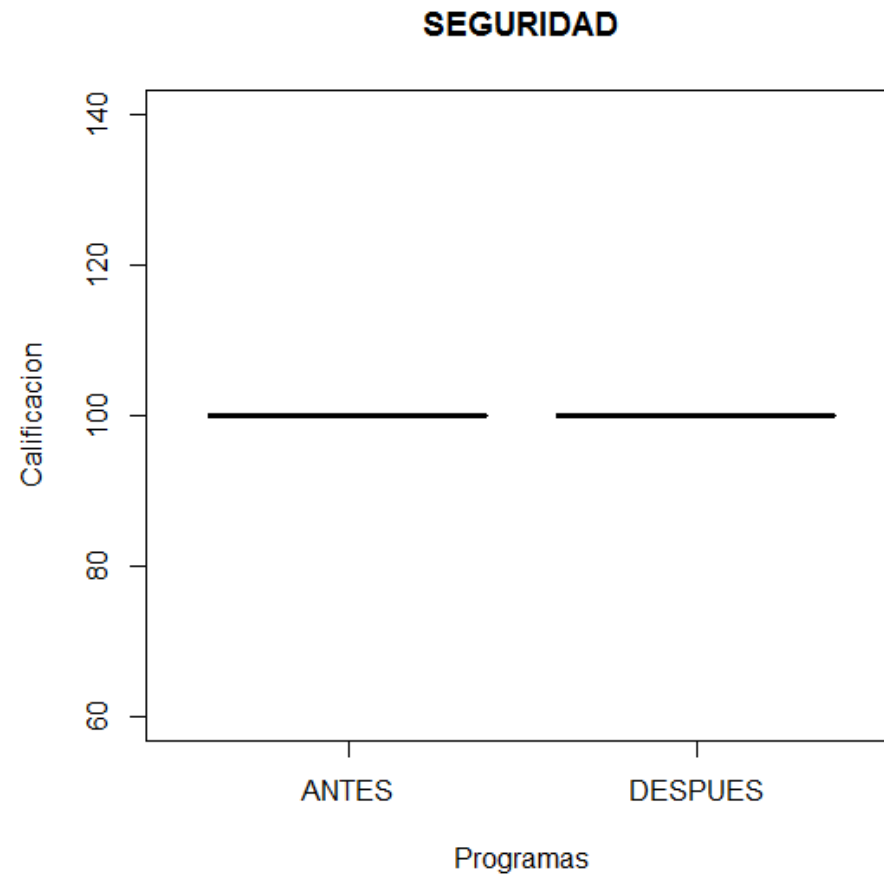


Figura66. Análisis de seguridad General.

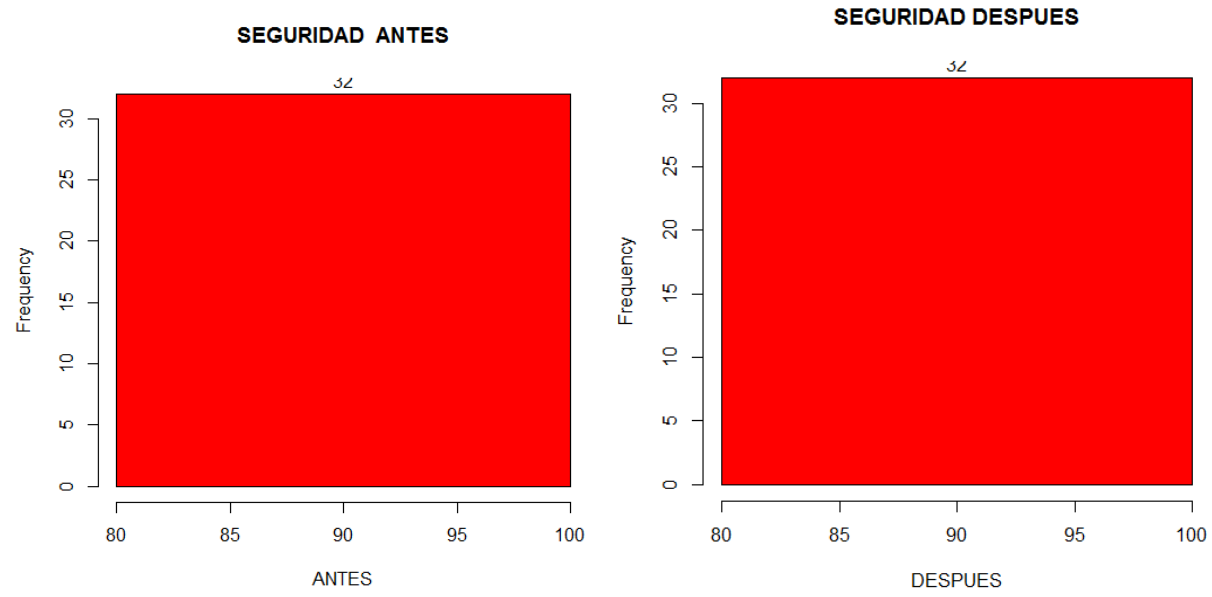


Figura67. Grafico de frecuencias pre y pos tes para variable seguridad.

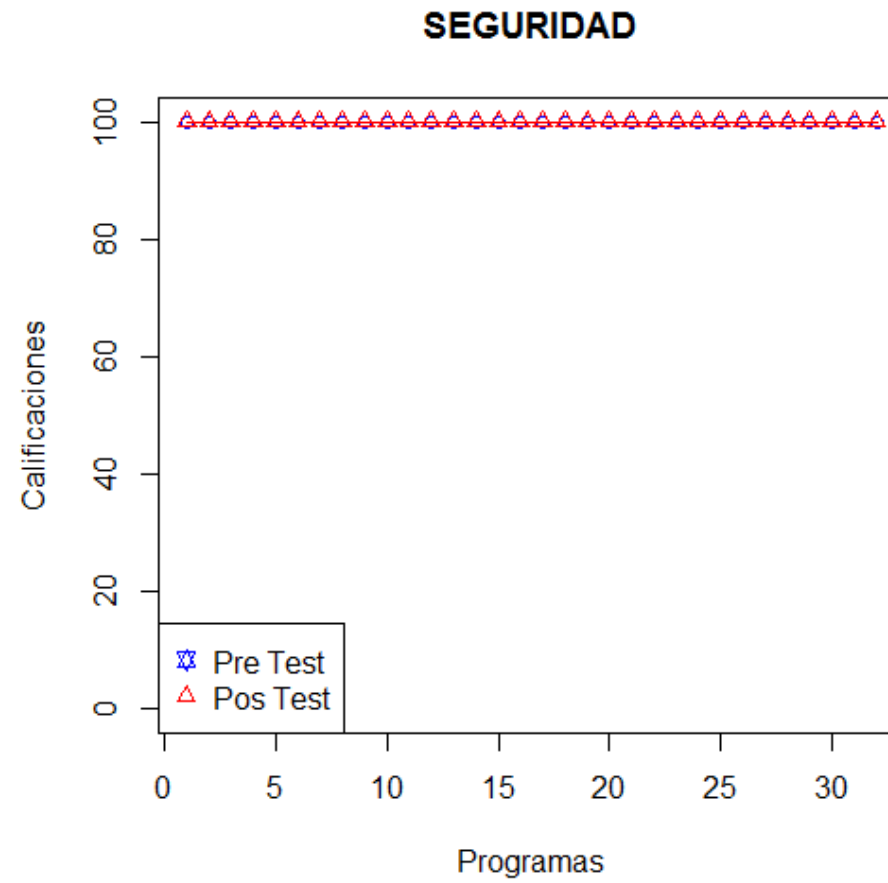


Figura68. Grafico de puntos pre y pos tes para variable seguridad.

Como se puede apreciar en la gráfica 65, la variable seguridad no varía en el post test, se mantiene igual debido a que los programas no tienen registro de datos para estadísticas de medición para seguridad. El grafico 66 muestra que la mediana para la variable seguridad es igual en el pos test, y pre test. El grafico 67 muestra las frecuencias con que aparecen las mediciones de seguridad calidad agrupadas en los rangos, donde en pre test y pos test los rangos con más frecuencias se mantienen iguales.

El grafico 68, corrobora la información mostrada en los anteriores, se muestra la tendencia constante de las calificaciones de los programas en pos test para la variable seguridad.

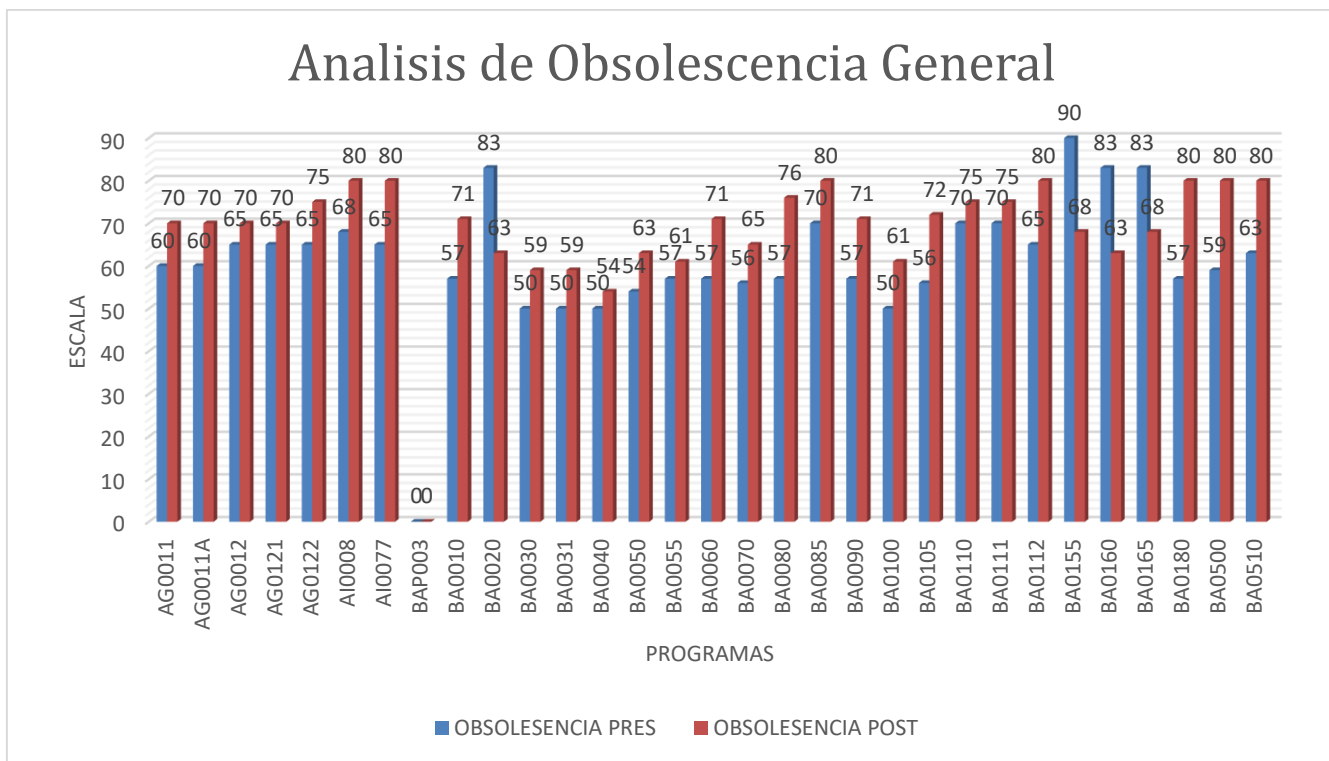


Figura69. Análisis de Obsolescencia General.

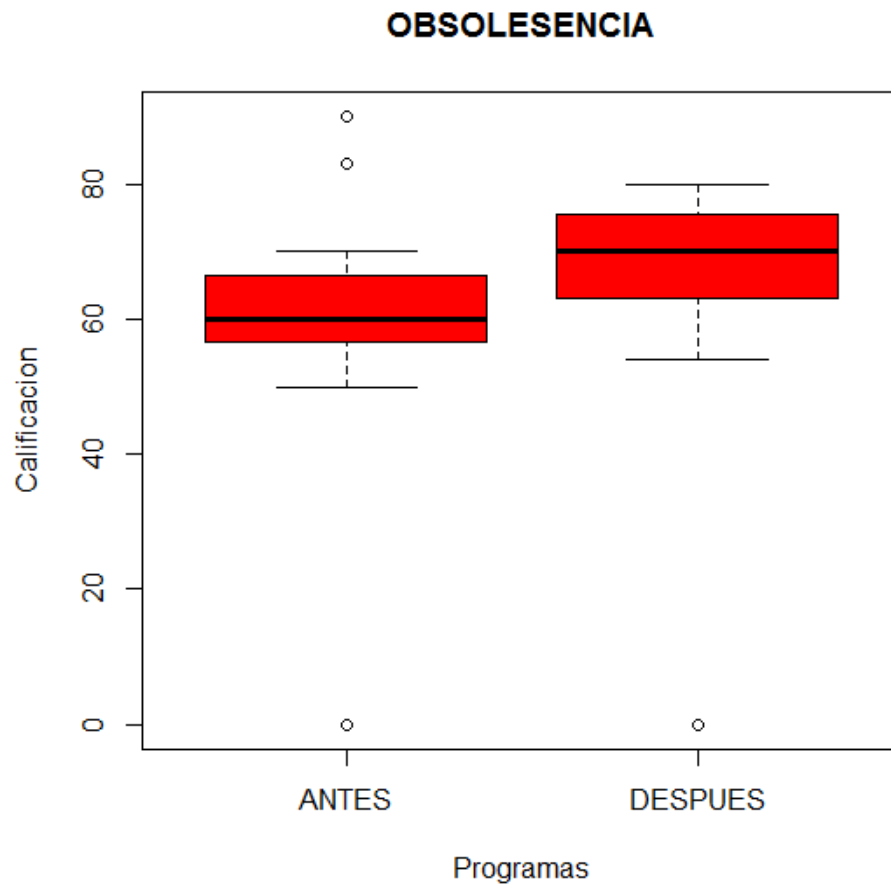


Figura70. Análisis de Obsolescencia General.

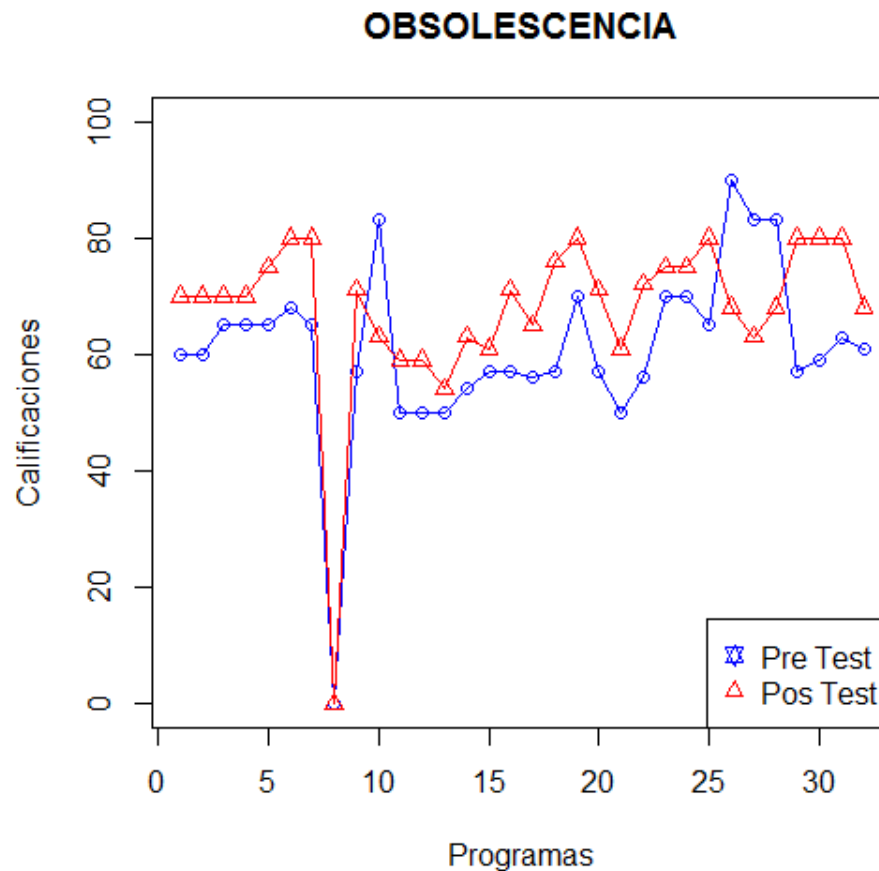


Figura71. Grafico de puntos pre y pos tes para variable obsolescencia.

Como se puede apreciar en la gráfica 69, existe una tendencia de crecimiento en los valores de la variable OBSOLESCENCIA medida en los programas analizados, lo que permite concluir que luego de una primera iteración de cambio o mejora en los fuentes de los programas, fruto de las observaciones de las métricas de OBSOLESCENCIA, se ha mejorado la GESTION DE OBSOLESCENCIA de las aplicaciones estos resultados apoyan la hipótesis general de la tesis de que el uso de la herramienta SOURCE400, mejora la GESTION DE OBSOLESCENCIA de las aplicaciones desarrolladas en lenguaje RPG.El grafico 70 muestra que la mediana

para la variable obsolescencia es mayor en el pos test, lo que muestra un promedio mayor de programas con calificación (75%) por encima de 80 contra valores inferiores a 60 del pre test. El grafico 71 muestra picos o conglomerados más altos en las barras 55-60 pre test, 75-80 post test que son los valores más comunes, la dispersión de los datos es desde 0 hasta 100. El histograma tiene una línea de distribución ajustada, sigue las alturas de las barras, los datos se ajustan a la distribución.

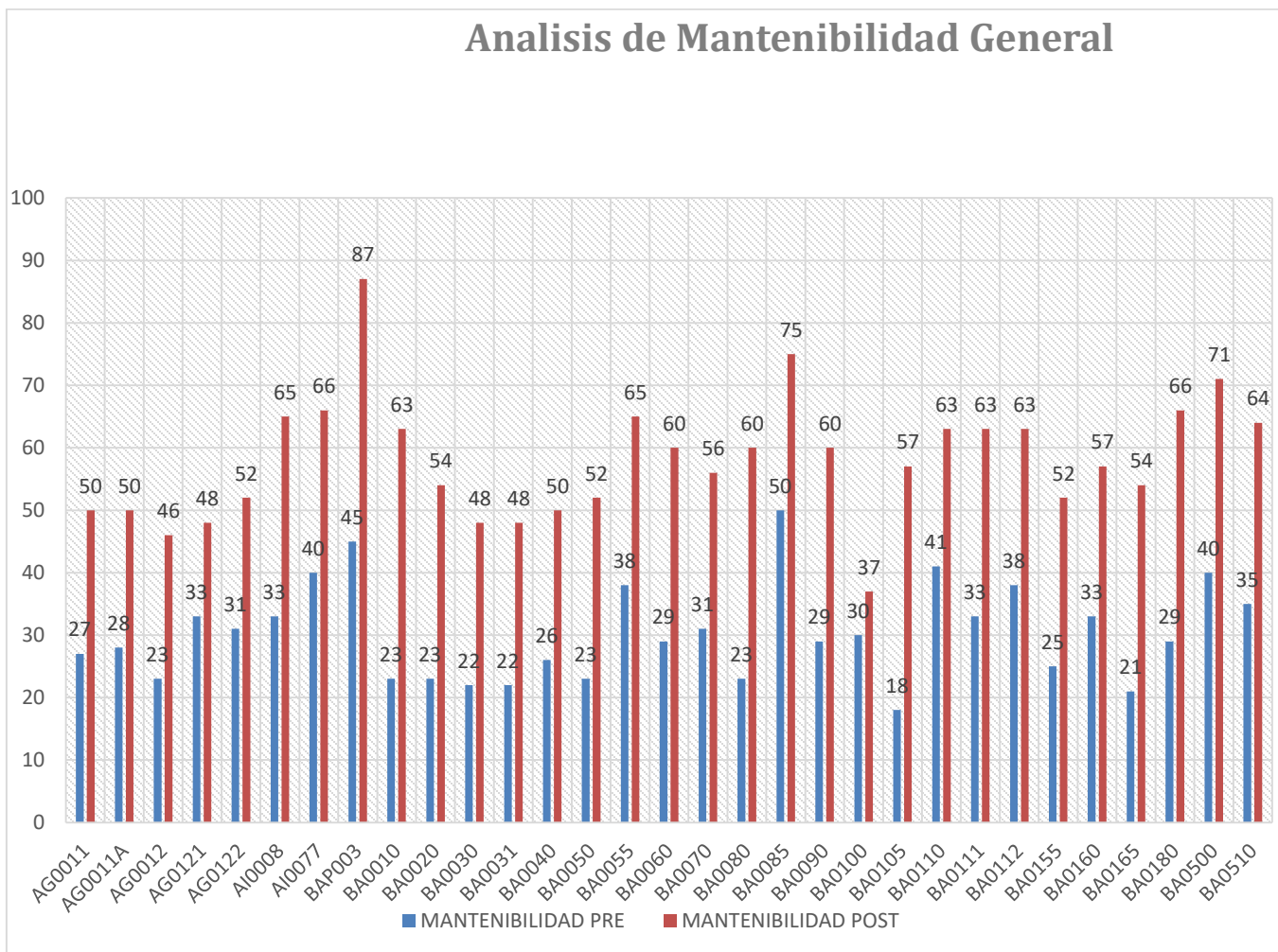


Figura72. Grafico de análisis de mantenibilidad general.

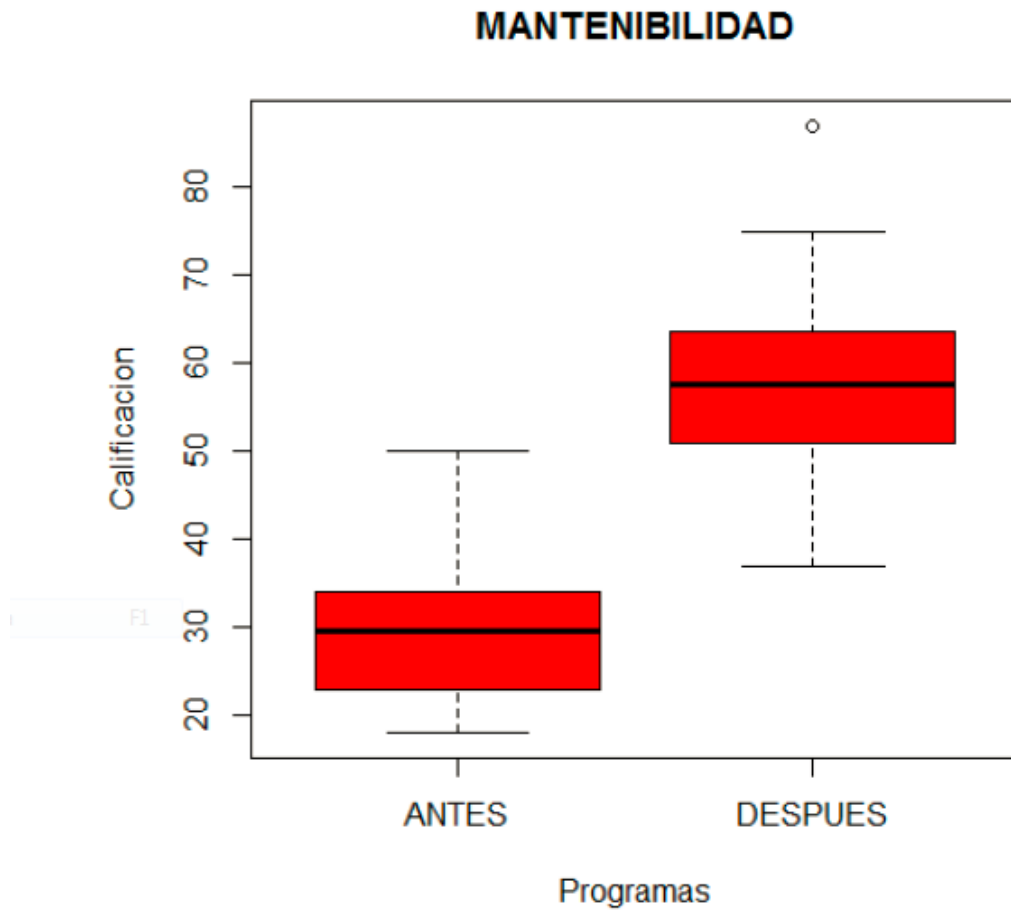


Figura73. Grafico boxplots de mantenibilidad general.

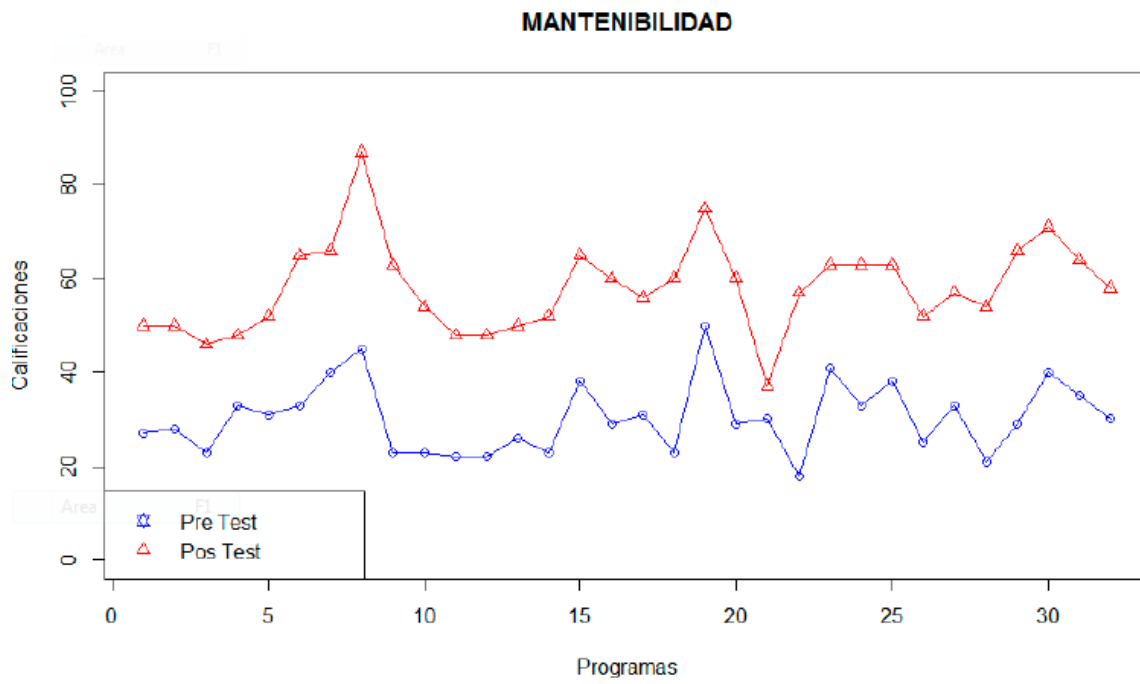


Figura74. Grafico de líneas de pre tes y post test de mantenibilidad general.

Como se puede apreciar en la gráfica72, existe una tendencia de crecimiento en los valores de la variable MANTENIBILIDAD medida en los programas analizados, lo que permite concluir que luego de una primera iteración de cambio o mejora en los fuentes de los programas, fruto de las observaciones de las métricas de MANTENIBILIDAD, se ha mejorado la GESTION DE MANTENIBILIDAD de las aplicaciones estos resultados apoyan la hipótesis general de la tesis de que el uso de la herramienta SOURCE400, mejora la GESTION DE MANTENIBILIDAD de las aplicaciones desarrolladas en lenguaje RPG.El grafico 73 muestra que la mediana para la variable obsolescencia es mayor en el pos test, lo que muestra un promedio mayor de programas con calificación (75%) por encima de 80 contra valores inferiores a 60 del pre test. El grafico 74 muestra picos o conglomerados más altos en las barras 55-60 pre test, 75-80 post test que son los valores más comunes, la dispersión de los datos es desde 0 hasta 100. El histograma tiene una línea de distribución ajustada, sigue las alturas de las barras, los datos se ajustan a la distribución.

3.13 Relación de variable Calidad con los indicadores

Debido a que las métricas de calidad de software según la IEEE 1061-1998, deben estar validadas, se analizará la relación de esta variable con las dimensiones para ver como aportan a la calidad total, para lo cual se utilizará, la prueba de correlación lineal. Inicialmente se hará una mirada general de las correlaciones entre las variables usando el gráfico de pares.

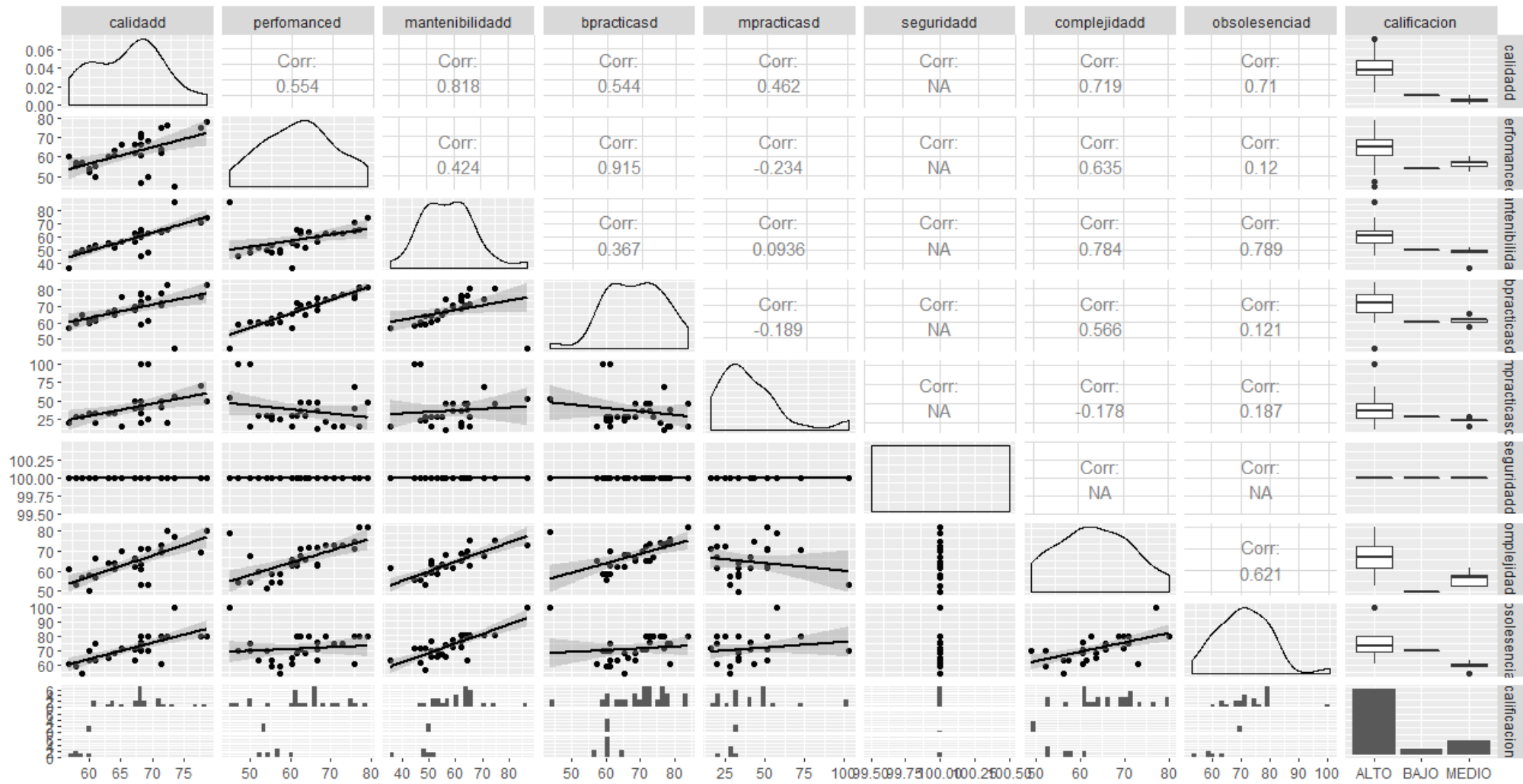


Figura 75. Scatterplot de calidad y sus dimensiones.

La figura 75 muestra un gráfico scatterplot múltiple para la variable calidad y sus indicadores mantenibilidad, buenas prácticas, malas prácticas, seguridad, complejidad y obsolescencia. La calidad tiene una fuerte correlación con los indicadores mantenibilidad, complejidad y obsolescencia observándose una relación positiva y una baja correlación con performance, buenas prácticas y malas prácticas. Seguridad no tiene correlación debido a su varianza 0. A si mismo se observa una correlación negativa entre los indicadores performance vs malas prácticas, buenas prácticas y malas prácticas.

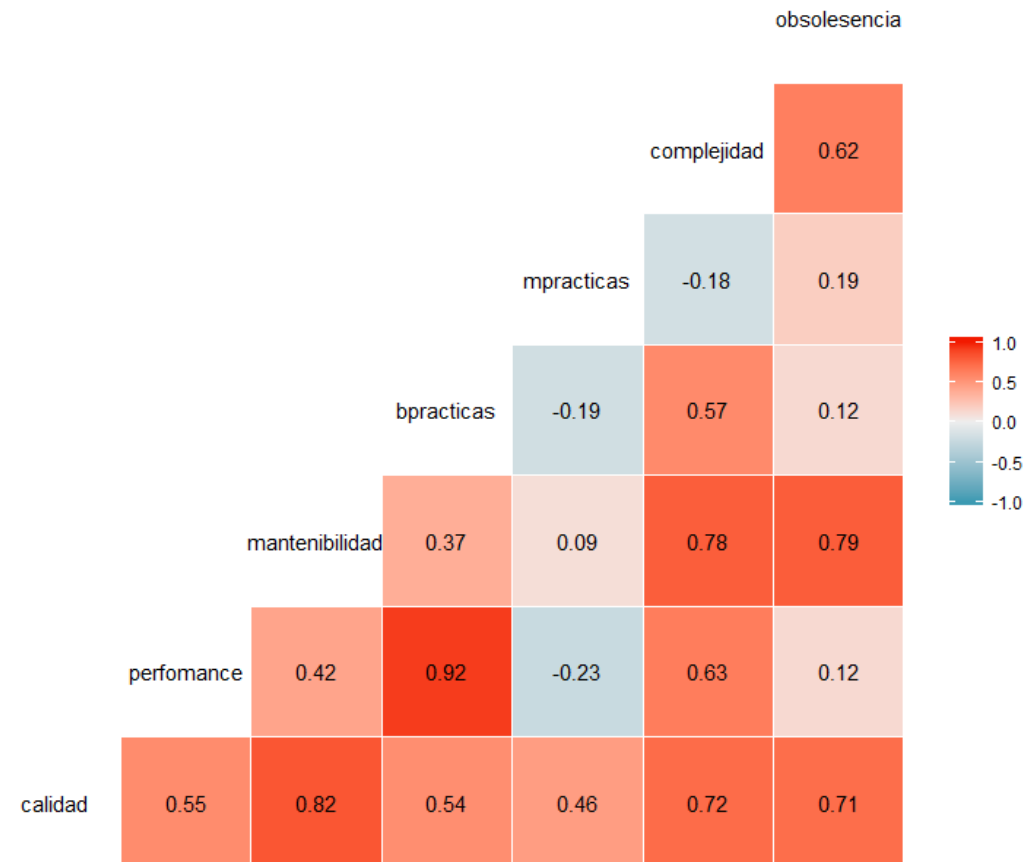


Figura76. Matriz de correlación de calidad y sus indicadores.

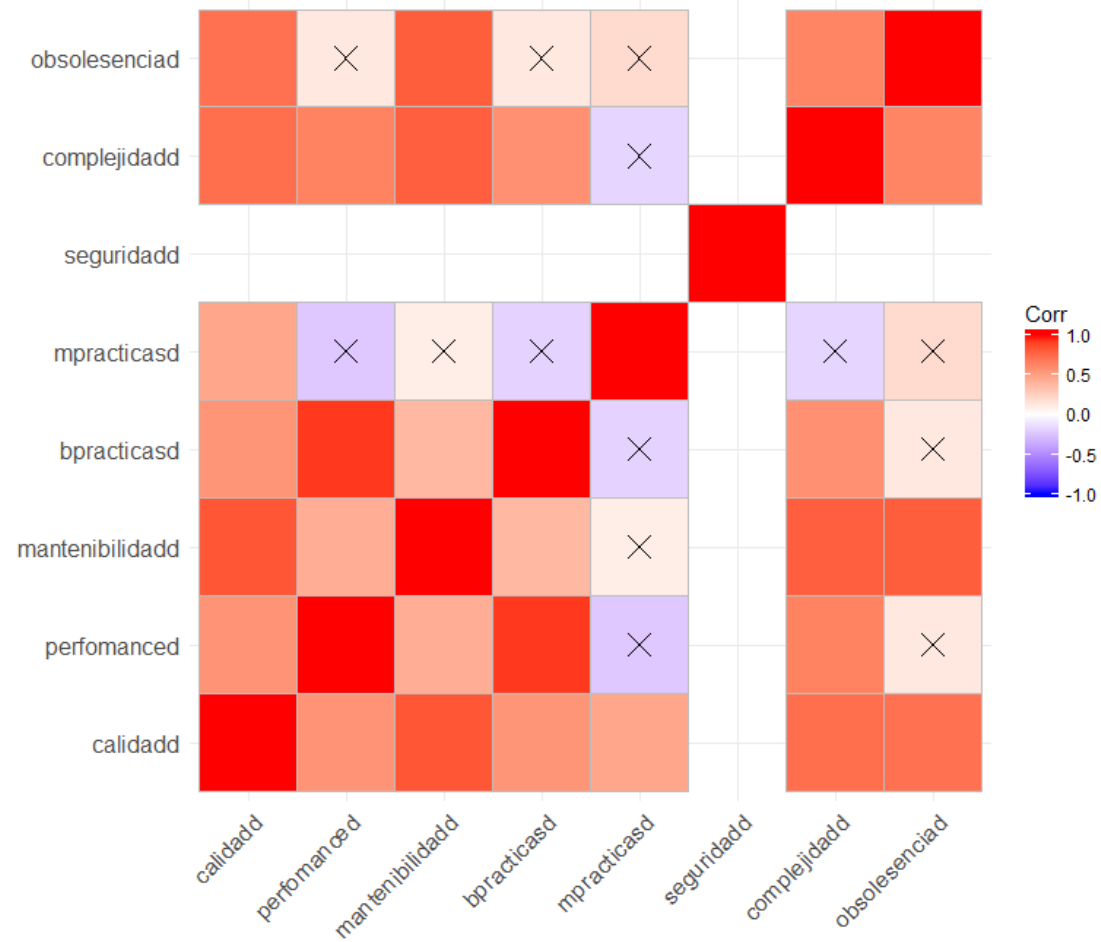


Figura 77. Matriz de correlación de calidad y sus indicadores.



Figura78. Matriz de correlación de calidad y sus indicadores.

En las figuras 76-78, se muestra la matriz de correlación para la variable calidad y sus indicadores performance, mantenibilidad, buenas prácticas, malas prácticas, seguridad, complejidad y obsolescencia. La calidad tiene una fuerte correlación con los indicadores mantenibilidad, complejidad y obsolescencia y una baja correlación con performance, buenasprácticas y malas prácticas.

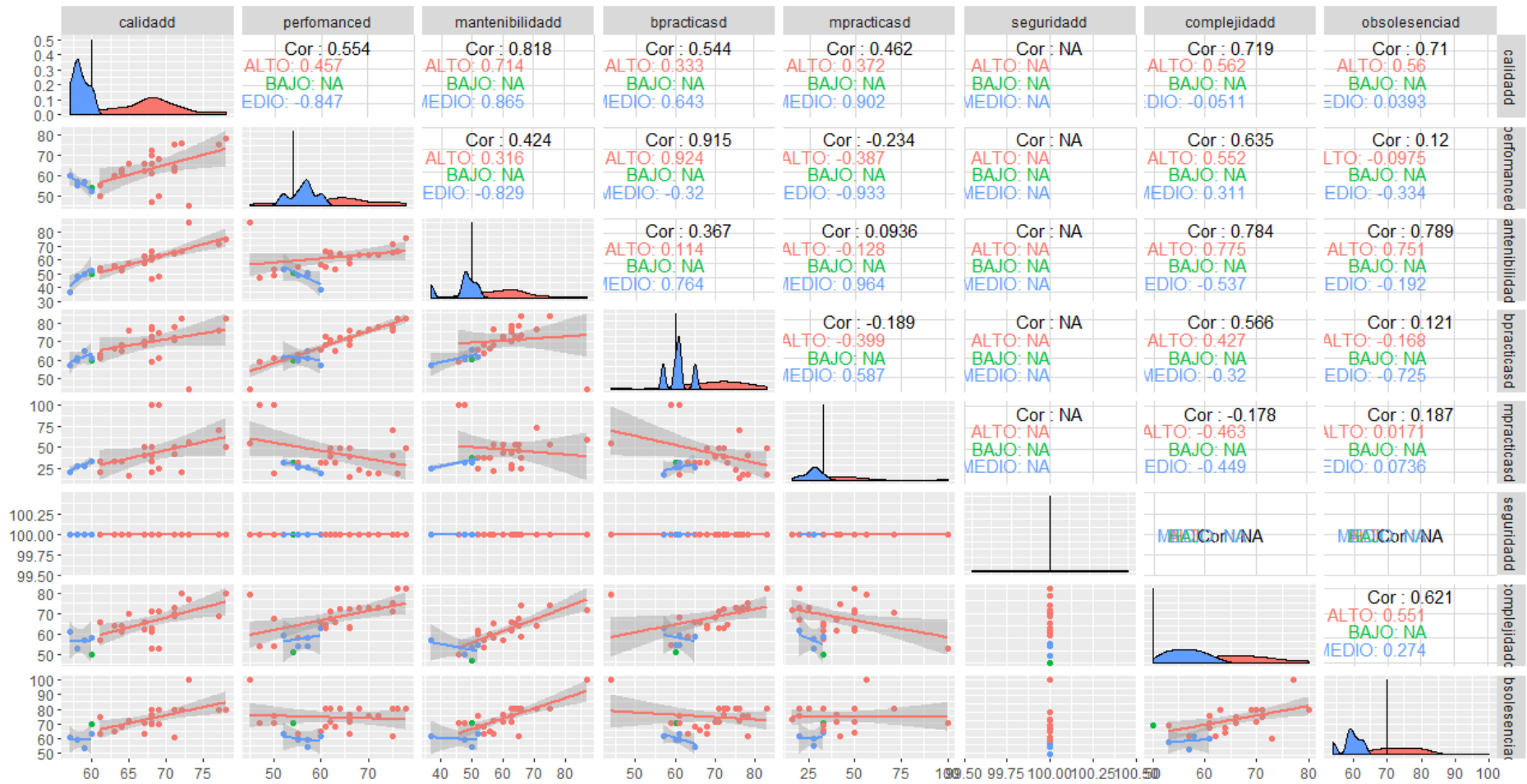


Figura79. Matriz de correlación entre calidad y las dimensiones.

La figura 79 muestra un graficoscatterplotmúltiple para la variable calidad y sus indicadores mantenibilidad, buenas prácticas, malas prácticas, seguridad, complejidad y obsolescencia. La calidad tiene una fuerte correlación con los indicadores performance, mantenibilidad, buenas prácticas, malas prácticas, complejidad y obsolescencia observándose una relación positiva y una baja correlaciónse presenta entre las dimensiones performance, mantenibilidad y buenas prácticas con malas prácticas observándose una relación negativa. Complejidad y malas prácticas también presenta una relación negativa. Performance, malas prácticas y buenas prácticas con obsolescencia presentan una relación negativa.

Seguridad no tiene correlación debido a su varianza 0. A si mismo se observa una correlación negativa entre los indicadores performance vs malas prácticas, buenas prácticas y malas prácticas.

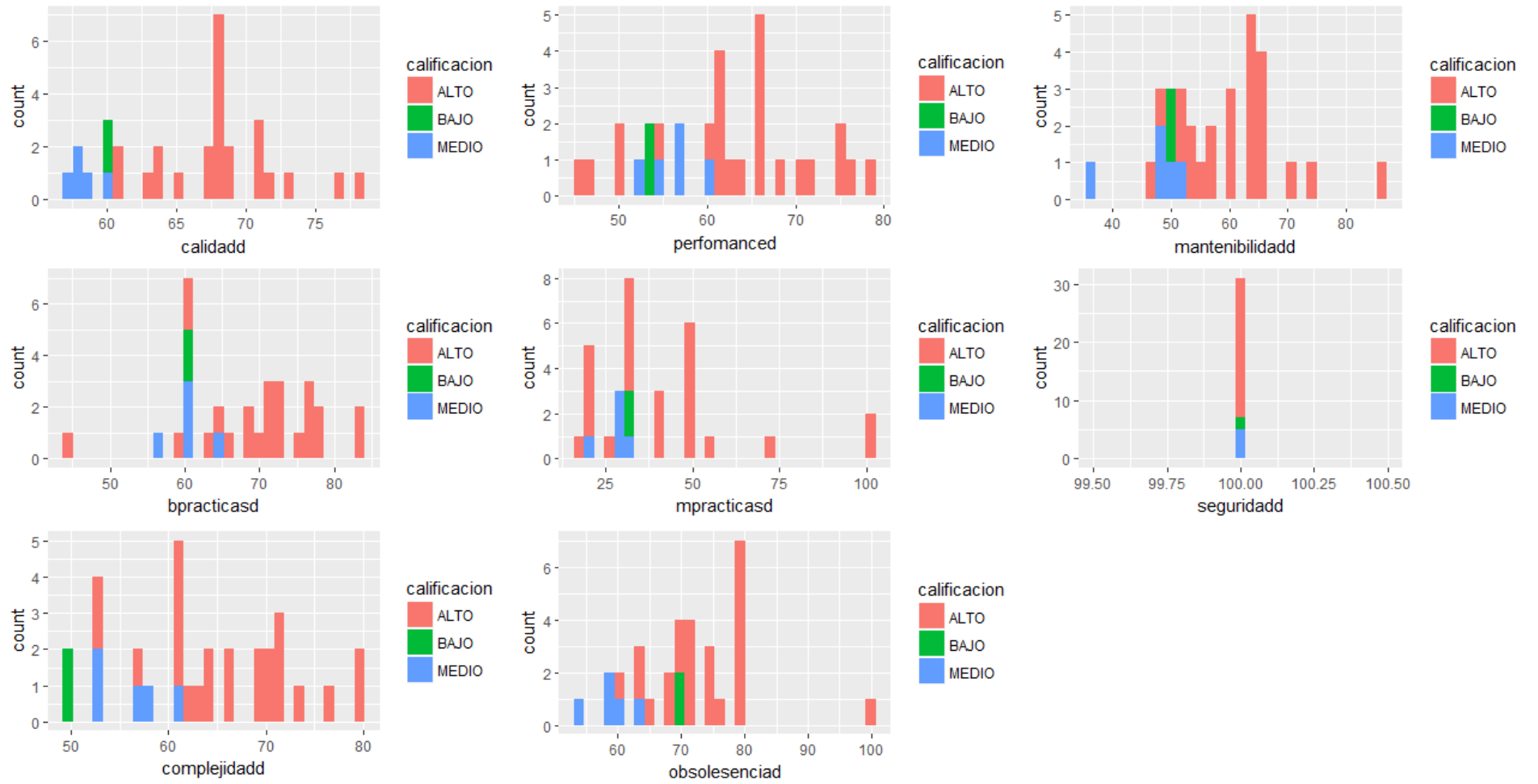


Figura80. Histogramas pos test.

La figura 80 muestra gráficos de barras que representan el número de calificaciones por rango, ya sea alto, medio o bajo, se observa un conteo mayor para calificaciones por encima de 60 en buenas prácticas, malas prácticas y obsolescencia; se observa una tendencia creciente para valores altos y medios, y una tendencia decreciente para valores bajos. Se observa mayor frecuencia de valores de calificación entre 70 – 80 para performance, buenas prácticas y complejidad.

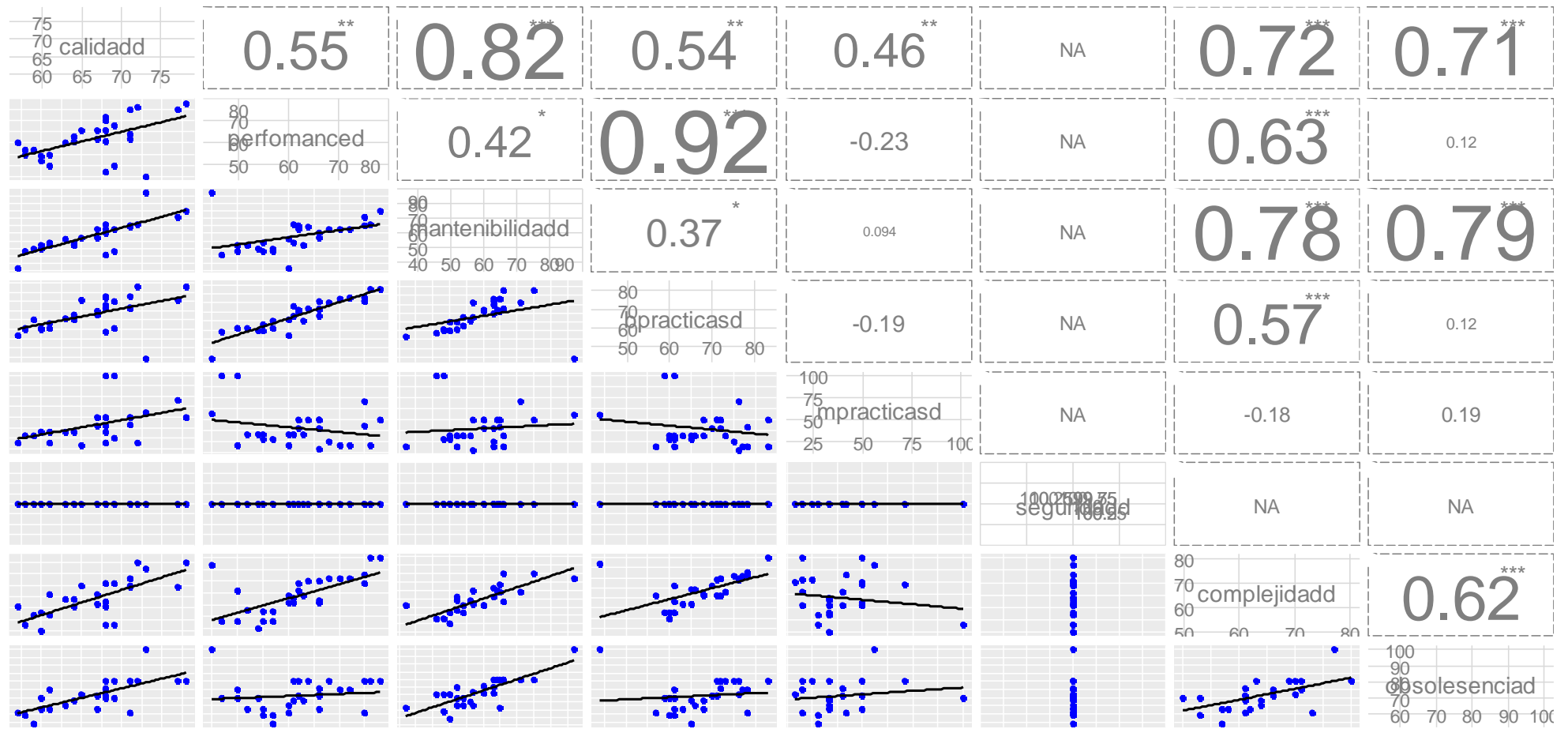


Figura81. Matriz de correlación entre calidad y las dimensiones.

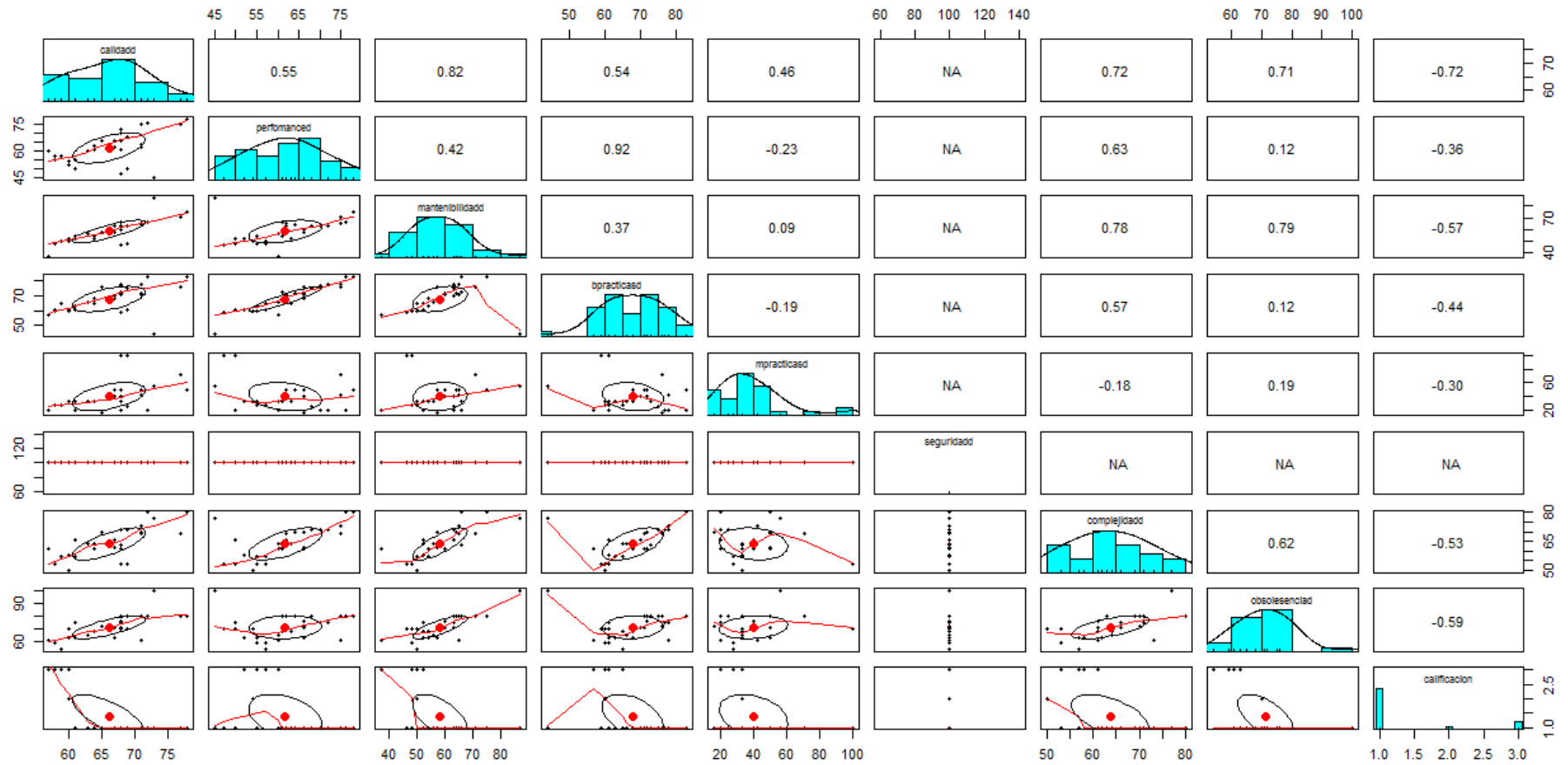


Figura82. Matriz de correlación entre calidad y las dimensiones.

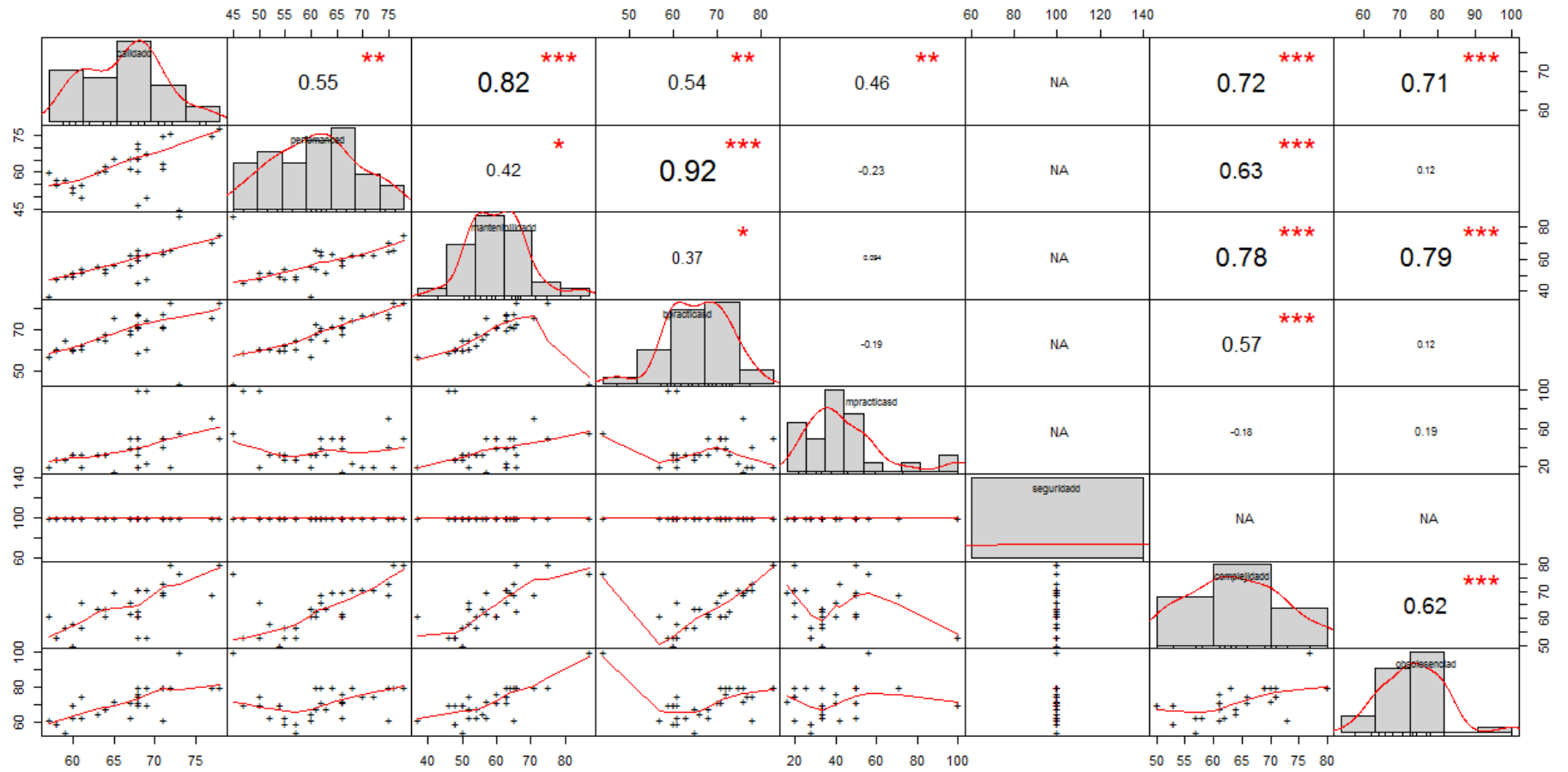


Figura83. Matriz de correlación entre calidad y las dimensiones.

Como se ve en los graficos 81-83, hay correlación fuerte entre performance y Mantenibilidad, performance y buenas prácticas, performance y malas prácticas, performance y complejidad, performance y obsolescencia, fuerte entre calidad y buenas prácticas, débil entre calidad y malas prácticas, débil entre calidad y complejidad, fuerte entre calidad y obsolescencia, débil entre buenas prácticas y malas prácticas, buenas prácticas y complejidad, fuerte entre buenas prácticas y obsolescencia , débil entre malas prácticas y complejidad , fuerte entre malas prácticas y obsolescencia, fuerte entre complejidad y obsolescencia.

La mayoría de las relaciones de los gráficos son monotonicas, o sea, si una variable sube o se incrementa, la otra también y se acercan bastante a la lineal. La tabla 113 muestra el cálculo de las correlaciones.

Tabla 113

Correlacion calidad e indicadores.

Calidad	performance	matenibilidad	bpractic	
Calidad	1.0000000	0.5536953	0.81758573	0.5439008
Performance	0.5536953	1.0000000	0.42378142	0.9152927
Mantenibilidad	0.8175857	0.4237814	1.00000000	0.3667171
Bpractic	0.5439008	0.9152927	0.36671710	1.0000000
Mpractic	0.4621058	-0.2344696	0.09357923	-0.1894412
Complejidad	0.7185997	0.6345827	0.78383064	0.5656289
Obsolescencia	0.7103490	0.1199948	0.78875354	0.1206501
	Mpractic	complejidad	Obsolescencia	
Calidad	0.46210584	0.7185997	0.7103490	
Performance	-0.23446957	0.6345827	0.1199948	
Mantenibilidad	0.09357923	0.7838306	0.7887535	
Bpractic	-0.18944116	0.5656289	0.1206501	
Mpractic	1.00000000	-0.1778224	0.1871412	
Complejidad	-0.17782241	1.0000000	0.6206217	
Obsolescencia	0.18714117	0.6206217	1.0000000	

IV Discusión

Ladányi, et al (2015) en su publicación, paper de la Investigación “A software quality model for RPG”, usaron un modelo de calidad que implementaron en una herramienta que midió la calidad de programas en RPG, el cual se baso en métricas no orientadas al lenguaje sino en métricas generales de ingeniería de software (Mantenibilidad, testabilidad, Analizabilidad, reusabilidad y modularidad), *figura 149*.

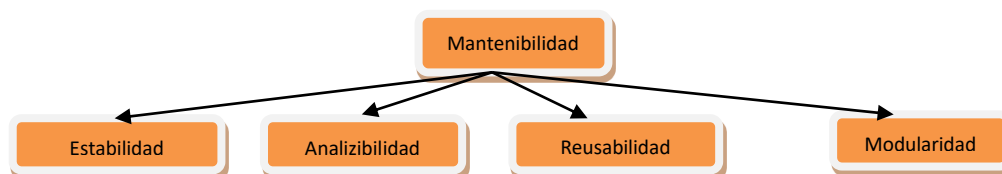


Figura 84. Modelo de calidad. Fuente Ladányi, et al (2015).

Mientras que en esta investigación he usado métricas orientadas al lenguaje (calidad, mantenibilidad, malas prácticas, buenas prácticas, seguridad, complejidad y obsolescencia). *Figura 150*.



Figura85. Modelo de calidad usado con QSOURCE.

Ladányi, et al (2015) obtuvieron resultados centrándose en la mantenibilidad mientras que en esta investigación se baso en la calidad como centro dependiendo de 6 dimensiones todas asociando características propias del lenguaje RPG, criterio que se concluye es un enfoque más preciso que el general orientado a métricas generales de ingeniería de software,

V.Conclusiones

- Primero** Luego de observar los resultados estadísticos, queda claro que el uso de una herramienta CASE (Computerasistant software Engineering) como QSOURCE para el análisis estático de la calidad del código desarrollado en lenguaje RPG es útil en ambientes de sistemas donde se requiera visibilizar la calidad para toma de decisiones. La herramienta QSOURCE permitió gestionar la calidad del software dentro del ciclo de vida de desarrollo de software del banco AGROBANCO. Esta investigación ha evidenciado que el uso de la herramienta QSOURCE en una primera iteración ha mejorado la calidad de los programas de la muestra, pero si se sigue usando en iteraciones posteriores, se podría llegar a un nivel superior de calidad.
- Segundo:** La presente investigación respecto al objetivo general, demuestra que el uso de QSOURCE como herramienta de gestión de calidad, ayudo a los gerentes de desarrollo y programadores, a controlar el perfomance, la mantenibilidad, las buenas prácticas, las malas prácticas, la complejidad, la seguridad y obsolescencia del código en programas desarrollados en lenguaje RPG de IBM, y ayudo a reducir el tiempo usado en poner en producción las aplicaciones.
- Tercero:** La presente investigación respecto al objetivo específico 1, ha determinado que el efecto de la implementación del Sistema QSOURCE en la mejora de la performance del software hecho en RPG en el banco Agrobanco, fue positivo lográndose demostrar la mejora en la calificación inicial de 53 a 61
- Cuarto:** La presente investigación respecto al objetivo específico 2, ha determinado que el efecto de la implementación del Sistema QSOURCE en la mejora de la mantenibilidad del software hecho en RPG en el banco Agrobanco, fue positivo lográndose demostrar la mejora en la calificación inicial de 30 a 58
- Quinto:** La presente investigación respecto al objetivo específico 3, ha determinado que el el efecto de la implementación del Sistema QSOURCE en el uso de buenas prácticas de codificación de Software en los programas hechos en RPG en el banco Agrobanco,

fue positivo lográndose demostrar la mejora en la calificación inicial de 48 a 67.

- Sexto:** La presente investigación respecto al objetivo específico 4, ha determinado que el efecto de la implementación del Sistema QSOURCE en la eliminación de malas prácticas de codificación de software hecho en RPG en el banco Agrobanco, fue positivo lográndose demostrar la mejora en la calificación inicial de 24 a 33
- Séptimo:** La presente investigación respecto al objetivo específico 5, ha determinado que el efecto de la implementación del Sistema QSOURCE en la mejora de la seguridad del software hecho en RPG en el banco Agrobanco, no se evidencio debido a que sus programas ya tenían la máxima calificación 100.
- Octavo:** La presente investigación respecto al objetivo específico 6, ha determinado que el efecto de la implementación del Sistema QSOURCE en la simplificación de la complejidad del software hecho en RPG en el banco Agrobanco, fue positivo lográndose demostrar la mejora en la calificación inicial de 54 a 63.
- Noveno:** La presente investigación respecto al objetivo específico 7, ha determinado que el efecto de la implementación del Sistema QSOURCE en la erradicación de la obsolescencia de códigos en el software hecho en RPG en el banco Agrobanco, fue positivo lográndose demostrar la mejora en la calificación inicial de 61 a 68.
- Decimo:** Mantenibilidad, complejidad y obsolescencia fueron las dimensiones que mas aportaron a la calidad según se muestra en la matriz de correlación entre la calidad y las dimensiones, lo cual debe usarse para afinar el modelo de análisis en futuros trabajos de investigación. Las dimensiones buenas prácticas, malas prácticas y seguridad no aportaron significativamente a la calidad según se observa en la matriz de correlación.

VI. Recomendaciones

- Primero:** Si bien se logro demostrar la efectividad del uso de QSOURCE en el banco AGROBANCO, como herramienta de mejora de la calidad del software en una primera y única iteración, se recomienda continuar con su uso continuo, ya que la base de datos tendría más información y con ella los análisis y tendencias serian más robustas y apegadas a la realidad.
- Segundo:** La empresa debe planear la creación de información histórica sobre la calidad y crear una data mart dentro de su datawarehouse, para así tener información a disposición de la gerencia de sistemas y desarrollo para la toma de decisiones. Se debe modificar el diagrama funcional para incluir un responsable de calidad.

VII. Referencias

- Agren, M (2009). *Static Code Analysis for Embedded Systems (Master of Science Thesis in Computer Science and Engineering)*. Department of Computer Science and Engineering Chalmers university of Gothenburg.
- Aho, A, Lam V, S, M. Sethi, Ravi. Ullman, Jeffrey D. (2007). *Compilers Principles Techniques, and Tools (2a Ed.)* Boston: Addison Wesley.
- Alvaro A & Almeida E.S. & Meira S.R.L. (2005). "Towards a Software Component Quality Model," Submitted to the 5th International Conference on Quality Software (QSIC).
- Artigas Morales Carles (2017). *Continuous integration and monitorization system for code quality and cooperation work*. Universidad
- Ahuja Pranjul. (2017). *Robust Malware Detection Using Integrated Static and Dinamic Analysis*. Indian Institute of Technology. India.
- Ayala Claudia & Hauge, Øyvind & Conradi Reidar & Franch Xavier & Li Jingyue. (2010). "Selection of third party software in Off-The-Shelf-based software Development—an interview study with industrial practitioners," *The Journal Of Systems and Software*, pp 24-36
- Badii, J. Castillo, M. Rodríguez, A. Wong P. (2007). *Diseños experimentales e investigación científica*. UANL, San Nicolás, N.L. 66450, México (s.e)
- Baldeón Villanes Edú James. (2015). *Método Para La Evaluación De Calidad De Software Basado En ISO/IEC 25000*. Universidad San Martin de Porres. Facultad De Ingeniería Y Arquitectura Sección De Posgrado. PERU.
- Ban Denes (2017). *Static Source Code Analysis in Pattern Recognition, Performance optimization and Software Maintainability*. Department of Software Engineering University of Szeged. Hungary.

- Bedoya, H, et al., 2014). *Modernizing IBM i Applications from the Database up to the User Interface and Everything in Between*. International Technical Support Organization. IBM RED BOOKS.
- Bertoa, M &Vallecillo A. (2002). "Quality Attributes for COTS Components," *I+D Computación*, Vol 1, Nro 2, 128-144.
- Binkley David (2007). *Source Code Analysis: A Road Map. Future of software Engineering*, 2007 FOSE 07.
- Bishal Shrestha (2016). *Best QMEs for measurement of Software quality for SMEs*. University of Eastern Finland.
- Bjerga Andreas (2016). *Automatic Test Case Generation for Go*. University Of Stavanger.
- Boehm, B. W., Brown, H., Lipow, M. (1978) "Quantitative Evaluation of Software Quality," TRW Systems and Energy Group, 1978
- Boulanger, J. (2013). *Static Analysis of Software*. Wiley. Retrieved from <http://www.ebrary.com>
- Carrasco Días, S. (2006). "Metodología de la Investigación científica". Editorial San Marcos. 1ra Reimpresión 2006. Lima. 2006.
- Chen Hongyu. (2014). "Analysis of Software Architecture Quality Metrics".Rwth AachenUniversity.Alemania.
- Dankovčíková Zuzana. (2017). "Custom Roslyn Tool for Static Code Analysis".MasarykUniversity.Faculty of Informatics.REPUBLICA CHECA.
- Dromey, R. G. (1995). "A model for software product quality". IEEE Transactions on Software Engineering, 21:146-162

Espejo Chavarría Alex José. (2016). "*Modelo de aseguramiento de la calidad en el ProcesodeDesarrollo de software basado en los modelos de madurez de Capacidades (CMMi), Proceso de software para equipos (TSP) y personas (PSP)*". FacultadDeIngenieríaDe Sistemas E InformáticaUnidadDePosgrado. PERÚ.

Espallargues Carreras Mireia, Almazán Sáez Cari, Pons Rafols Joan MV, erra Prat Mateu S. (2004). "*Evaluación en servicios sanitarios Análisis de la evidencia científica*". Universidad Oberta de Catalunya UOC. Barcelona UOC.

Georgiadoui, Elli "*GEQUAMO-A Generic, Multilayered, Customizable Software Quality model*". Software Quality Journal, 11, 4, 313-323.DOI=10.1023/A:1025817312035.

GordieievOleksandr (2014)."*Evolution of Software Quality Models in Context of the Standard ISO 25010*". Sevastopol Institute of Banking of University of Banking of the National Bank of Ukraine.

Grady, R. B. (1992). "*Practical Software Metrics for Project Management and Process Improvement*". Prentice Hall, Englewood Cliffs, NJ, USA

Hegedus Peter (2014). "*Advances in Software Product Quality Measurement and its Applications in Software Evolution*".Department of Software Engineering University of Szeged. Hungary.

Hernández, F. & Sánchez, J. (1998). "*Para enseñar no basta con saber la asignatura*".España: Ediciones Paidós Ibérica SA

Hernández, R. Fernández, C. & Baptista, L. (2010). "*Metodología de la investigación*". (5ª Ed.) México: Mc. Graw-Hill/ Interamericana.

Hietala Iiro. (2014). "*Assisting Software Quality Assurance byChange Impact Analysis*". JAMK University of Applied Sciences.FINLANDIA.

Hitz, Martin & Montazeri, Behzad. (1996). "*Chidamber and Kemerer's metrics suite: A measurement theory perspective*". Software Engineering, IEEE Transactions on. 22. 267 - 271. 10.1109/32.491650.

Horch, J. W. (2006). "*Software quality Engineering/Testing, quality assurance, and Quantifiable Improvement*". Milwaukee: American Society for quality.

Huanca Luis (2015). "*Revisión sistemática de la calidad del software en prácticas Ágiles*". Pontificia universidad católica del Perú. Perú.

Ivo Gómez, Pedro Morgado, Tiago Gomes, Rodrigo Morria (2011). "*An Overview On The Static Code Analysis approach in software development*". Universidad Do Porto.

IEEE Stándars Association. (2016). "*1061-1998 - IEEE Standard for a Software Quality Metrics Methodology*". Recuperado de: <https://standards.ieee.org/findstds/standard/1061-1998.html>

ISO/IEC. ISO/IEC 9126. "*Software Engineering – Product quality*". ISO/IEC, 2001.

ISO/IEC. ISO/IEC 25000:2005. *Software Engineering – Software product Quality Requirements and Evaluation (Square) – Guide to Square*. ISO/IEC, 2005.

ISO/IEC IS 9126-1. (2001). "*Software Engineering - Product Quality – Part 1: Quality Model*". International Organization for Standarization, Geneva, Switzerland.

ISO/IEC TR 9126-2. (2003). "*Software Engineering - Product Quality - Part 2: External Metrics*". International Organization for Standardization, Geneva, Switzerland.

ISO/IEC TR 9126-3. (2003): "*Software Engineering - Product Quality - Part 3: Internal Metrics, International Organization for Standardization*". Geneva, Switzerland.

- ISO/IEC TR 9126-4. (2004). "*Software Engineering - Product Quality - Part 4: Quality in Use Metrics*". International Organization for Standardization". Geneva, Switzerland.
- ISO/ IEC CD 25010. (2008). "*Software Engineering: Software Product Quality Requirements and Evaluation (SQuaRE) Quality Model and guide*". International Organization for Standardization, Geneva, Switzerland.
- IT Jungle (2016). IBM i Installed Base Dominated By Vintage Iron. Recuperado de: <http://www.itjungle.com/tfh/tfh120913-story01.html>
- IT Jungle (2016). The Shape of the System i Business. Recuperado de: <http://www.itjungle.com/tfh/tfh041006-story05.html>
- José P. Miguel, David Mauricio and Glen Rodríguez (2014). "*A Review of Software Quality Models for the Evaluation of Software Products*". Department of Exact Sciences, Faculty of Sciences, Universidad Peruana Cayetano Heredia, Lima, Peru.
- Klas Michael & Constanza Lampasona & Jurgen Munch. (2011). "*Adapting Software Quality Models: Practical Challenges, Approach, and First Empirical Results*," 37th EUROMICRO Conference on Software Engineering and Advanced Applications, 978-0-7695-4488-5/11, IEEE pp. 341-348
- Ladányi G, Tóth Z, Ferenc R and Keresztesi T. (2015) "*A software quality model for RPG*". *IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Montreal. 91-100. doi: 10.1109/SANER.2015.7081819
- McCall, J. A., y J. P. Cavano. (1978). "*A Framework for the Measurement of Software Quality*". ACM Software Quality Assurance Workshop.

- Mc Call, J. A. &, Richards, P. K. & Walters, G. F. (1977). "*Factors in Software Quality*". Volumes I, II, and III. US Rome Air Development Center Reports, US Department of Commerce, USA.
- Mena, Mendoza, G. (2006). ISO/9126-3 Métricas internas de calidad del producto de software. Recuperado de http://www.mena.com.mx/gonzalo/maestria/calidad/presenta/iso_9126-3/
- McGarry J., Card D., Jones C., Layman B., Clark E., Dean J., Hall F. (2001) "*Practical Software Measurement: Objective Information for Decision Makers.*", Addison-Wesley Professional.
- Morris, B. (2013). Formatos RPG. Recuperado de: ["http://www.tug.ca/MoMHandouts/2013-11-0/Free_Form_RPG_TUG_Nov_2013.pdf"](http://www.tug.ca/MoMHandouts/2013-11-0/Free_Form_RPG_TUG_Nov_2013.pdf)
- Murillo, W. (2008). "*La Investigación Científica*. Universidad Nacional de Colombia". Instituto de Inmunología de Colombia. Colombia.
- Namakforoosh, MohammadNaghi. (2005). "*Metodología de la Investigación*". (2aEd.) Mexico: Limusa.
- Papa María Fernanda.(2012)."*Aseguramiento de Calidad de Software: Estudio Comparativo de Estrategias de Medición y Evaluación*". Facultad de InformáticaUniversidad Nacional de LaPlata.Argentina.
- Pressman, R. S. (2010). "*Ingeniería del software: Un enfoque práctico*". (Séptima. ed.)
- Ramos Palacios DianaEstefanía.(2016)."*Diseño De Un Modelo De La Calidad DeProductosDeSoftware Basado EnMétricas Externas Y Usabilidad Aplicado A Un Caso De estudio*".FacultadDeIngeniería Y ArquitecturaSección De Posgrado.EscuelaPolitécnicaNacional.FacultadDeIngenieríaDeSistemas. Ecuador.

- RakicGordana (2015). "SSQSA set of software quality static analysers".universityOf Novi sadfaculty of sciencesdepartmentofmathematics and informatics.Serbia.
- Rawashdeh A, &MataalkahBassem. (2006). "A New Software Quality Model forEvaluating COTS Components," Journal of Computer Science 2 (4): 373-381, 2006
- Schulmeyer, G. G. (2008; 2007; 2015 ;). "*Handbook of software qualityAssurance (4thd.)*".GB: Artech House Inc.Senevirathne, A et al.(2012). Metric Based Code Analyzing Tool.
- S. R. Chidamber and C. F. Kemmerer, "A Metrics Suite for Object-Oriented Design", IEEE Trans. Software Eng., vol. 20, no. 6, June 1994, pp. 476-493.
- Taipale Taneli. (2015). "*Improving Software Quality withSoftwareError Prediction*". University ofOulu.Finlandia.
- Tian, J. (2007). "*Software quality engineering. Testing, Quality Assurance and QuantifiableImprovement*". New Jersey. E.U. John Wiley&Sons.
- Torres, B. (2007). "*Metodología de la investigación científica*". Universidad Nacional Mayor de San Marcos. Perú: San Marcos.
- Torres, G. (2012). "*Los cuatro periodos del periodo pe Piaget*". UNID. Recuperado de: http://moodle.unid.edu.mx/dts_cursos_md/maestria_en_educacion/teo_aprendiz_instruc/documentos/DesarrolloPiaget2.pdf.
- Williams, L. 2006. White-Box Testing. Retrieved from<http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf>

VIII. Anexos

Anexo 1

ARTÍCULO CIENTÍFICO

1. TÍTULO

QSOURCE en la calidad del software desarrollado en RPG

2. AUTOR

Giovanni Barrero Ortiz gbarreroo@hotmail.com Estudiante del Programa de Maestría en Gestión de Tecnología de Información de la Escuela de Postgrado de la Universidad Cesar Vallejo.

3. RESUMEN

Esta Investigación pre-experimental de una muestra tuvo como propósito fundamental determinar la influencia del uso del software QSOURCE en la mejora de la calidad del software desarrollado en lenguaje RPG en el departamento de sistemas del banco AGROBANCO. A una muestra de 31 programas se les analizó con la herramienta QSOURCE y su algoritmo de análisis estático de código generando calificaciones para cada uno de los 7 índices de la variable.

La información para el análisis y la recolección de datos se realizó a través de un pre-test, y un pos-test usando la herramienta QSOURCE. El análisis estadístico se realizó aplicando el SPSS a un grupo experimental antes y después del tratamiento. También se le aplicó la prueba T de Student al grupo antes y después de la estrategia para comprobar la equivalencia inicial de ellos. La base teórica de la variable calidad del software se sustentó en las técnicas de revisión y aseguramiento de calidad de software de Pressman (2010), en el modelo de calidad del software ISO/IEC 2510 y en IEEE 1061-1998 - IEEE Standard for a Software Quality Metrics. La base teórica para la variable sistema QSOURCE se sustenta en las técnicas de revisión y aseguramiento de calidad de software de Pressman (2010), en el modelo de calidad de software para RPG de Ladányi, et al (2015), y en el soporte de métricas de software automatizado de Senevirathne, et al (2012). Los resultados obtenidos evidenciaron una mejora en la calificación promedio de la calidad de 54.48 a 66.16 en una primera iteración.

Se compararon los resultados para el grupo con pre y el pos-test permitiendo demostrar la hipótesis del trabajo y la efectividad del software para mejorar la calidad, a través del uso del software los programas desarrollados en RPG incrementaron sus calificaciones para los índices de performance, mantenibilidad, buenas prácticas, malas prácticas, complejidad y obsolescencia, como se evidencia en los resultados.

4. PALABRAS CLAVE

Palabras claves: Análisis estático de código, RPG, Calidad de software, CASE, Software.

5. ABSTRACT

This pre-experimental investigation of a sample had as its fundamental purpose to determine the influence of the use of the QSOURCE software in the improvement of the software quality developed in RPG language in the systems department of the AGROBANCO bank. A sample of 31 programs was analyzed with the QSOURCE tool and its static code analysis algorithm, generating ratings for each of the 7 indexes of the variable.

The information for the analysis and data collection was made through a pre-test, and a post-test using the QSOURCE tool. The statistical analysis was performed applying the SPSS to an experimental group before and after the treatment. The Student's T test was also applied to the group before and after the strategy to check the initial equivalence of them. The theoretical basis of the software quality variable is based on Pressman's software quality review and assurance techniques (2010), the ISO / IEC 2510 software quality model and IEEE 1061-1998 - IEEE Standard for a Quality Metrics Software The theoretical basis for the QSOURCE system variable is based on Pressman's software quality review and assurance techniques (2010), on the RPG software quality model of Ladányi, et al (2015), and on the metrics support of automated software from Senevirathne, et al (2012). The results obtained showed an improvement in the average quality rating

from 54.48 to 66.16 in a first iteration.

The results were compared for the group with pre- and post-test allowing to demonstrate the work hypothesis and the effectiveness of the software to improve the quality, through the use of the software the programs developed in RPG increased their ratings for the performance indexes, maintainability, good practices, bad practices, complexity and obsolescence, as evidenced by the results.

6. KEYWORDS

Keywords: Static code analysis, RPG, Software quality, CASE, Software.

7. INTRODUCCIÓN

En ambientes bancarios en todo el mundo y también en el Perú (banco Agrobanco, banco Banbif, banco Financiero entre otros), donde se usa la plataforma IBM/AS400 se tiene un problema de calidad en las aplicaciones legadas que están desarrolladas en lenguaje RPG que vienen desde la década de 1959 y tienen versiones antiguas del lenguaje donde el software desarrollado en los años subsecuentes sigue los lineamientos antiguos y obsoletos, este problema se ha venido dando hace mucho tiempo atrás por el desconocimiento de técnicas de Ingeniería de Software para mejorar las millones de líneas de código antiguo de las aplicaciones y por no existir herramientas de análisis estático que analicen características únicas del lenguaje RPG relacionadas con calidad.

Los millones de líneas de código de programas hechos en RPG actualmente en los bancos que usan IBM/RPG, tienen problemas de performance, al mantener códigos que hacen uso ineficiente de los recursos de CPU y memoria, igualmente existen código que impide la mantenibilidad de esos programas, existen códigos en desuso apuntando a malas prácticas, tienen pocas buenas prácticas de codificación implementadas, usan código y técnicas que hacen muy complejo los sistemas y poseen código con potenciales problemas de seguridad.

Ladányi, et al (2015) publicaron un paper de la Investigación "A software quality model for RPG". In 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER) (pp. 91-100). IEEE. Los

investigadores crearon un modelo de calidad basado en el estándar ISO/IEC25010. La principal contribución de este estudio es la creación de un enfoque de gestión de calidad continuo especializado para RPG. Investigaron el estado del arte en las técnicas de evaluación de calidad para RPG, para explorar el código RPG y detectar componentes con alto riesgo, para ello usaron la herramienta SourceMeter para análisis de código estático. El analizador proveyó 3 tipos de medición: métricas de software, violación de reglas y duplicación de código.

Senevirathne, et al (2012) publicaron un paper: *Metric Based CodeAnalyzingTool*. El objetivo principal de la publicación fue proveer un soporte de métricas de software automatizado para usuarios que analicen código fuente de software. Esta herramienta evalúa la calidad del software y formula estimaciones de acuerdo a un modelo basado en métricas jerarquizadas especificadas, que genera comentarios de calidad, análisis de resultados, revisión y resolución basados en resultados de métricas. Actualmente no hay un modelo estructurado para medir el código fuente que sea predictivo y confiable, sin embargo, existen muchas herramientas de análisis de código para asistir en la medición de software en la industria, la mayoría de ellas simplemente recolecta métricas no estructuradas, pero no proveen un enfoque bien definido para relacionar esas métricas a atributos externos de calidad del software.

QSOURCE es una herramienta de software que permite a los ingenieros de software, desarrolladores y gerentes de desarrollo, visibilizar y cuantificar la calidad de sus sistemas desarrollados en RPG, y así poder tomar decisiones que les permitan alinear sus procesos a los objetivos del negocio, en cuanto a la generación de valor. QSOURCE está desarrollado en lenguaje RPG, y funciona en la plataforma IBM/AS400 o I SERIES, permite definir métricas de calidad a validar en las fuentes de los programas evaluados, se definen las librerías donde se encuentran las fuentes de programas RPG (RPG III, IV, FREE, SQLRPGLE, RPGLE) y se procede a analizar las fuentes de manera manual uno a uno o de manera masiva, al final se genera los resultados del análisis ya sea por pantalla o por reporte.

```

SOU000008 SISTEMA2 SOURCE/400 RESUMEN ESTADISTICA DL6334 ..... 8/01/16
Performance 19 / 69 Calidad 17 / 86
Best Pract. 18 / 86 Bad Pract. 4 / 24
Seguridad / 6 Complejidad 11 / 45
Obsolesc. / 18
OP METRICA VALOR . . PARAMETROS. VLR. INI. VLR. FINAL
--- IMPORT S S S N N S N 1 100
--- CALL 27 S S S N N S N 1 20
--- CALLB S S S N N S N 1 20
--- EXTPROC S S S N N S N 1 20
--- CALLP 1 S S S N N S N 1 20
--- GOTO S S S N N S S 1 10
--- EXSR 107 S S S N N S N 1 50
--- IF 367 S S S N N S N 1 10
--- DOW 23 S S S N N S N 1 10
--- DOU S S S N N S N 1 10
F3=Salir
    
```

```

SOU8888848 Consulta General de Calificaciones Fecha.: 8/01/17
Usuario: RGRGBD
Libreria..... Fecha:
Archivo Fuente..... Hora:
Programa.....
SI Libreria A Fuente Programa Fecha Hora Perfo Calidad B.prac N.Prac Segur. Complej. ObsoL
- L16680 FUENTES1 A68811 2817-85-88 14.57.88 54 33 47 27 188 58 68
- L16680 FUENTES1 A68810 2817-85-88 14.57.88 54 34 50 27 188 58 68
- L16680 FUENTES1 A68812 2817-85-88 14.57.88 47 30 47 188 53 65
- L16680 FUENTES1 A68821 2817-85-88 14.57.88 50 39 57 188 53 65
- L16680 FUENTES1 A68822 2817-85-88 14.57.88 50 31 46 14 188 66 65
- L16680 FUENTES1 A18888 2817-85-88 14.57.88 53 46 59 75 188 58 68
- L16680 FUENTES1 A18877 2817-85-88 14.57.88 61 40 61 16 188 63 65
- L16680 FUENTES1 B88893 2817-85-88 14.57.88 188 188 50 188 188
- L16680 FUENTES1 B88810 2817-85-88 14.57.88 42 33 35 39 188 46 57
- L16680 FUENTES1 B88820 2817-85-88 14.57.88 45 28 38 33 188 46 83
- L16680 FUENTES1 B88830 2817-85-88 14.57.88 47 27 48 33 188 46 58
- L16680 FUENTES1 B88831 2817-85-88 14.57.88 49 27 42 33 188 46 58
- L16680 FUENTES1 B88840 2817-85-88 14.57.88 52 31 47 37 188 58 50
- L16680 FUENTES1 B88850 2817-85-88 14.57.88 44 29 37 42 188 46 54
- L16680 FUENTES1 B88855 2817-85-88 14.57.88 63 38 50 28 188 64 57
F3=Salir F18=Ayuda
    
```

Figura 1. QSOURCE

8. METODOLOGÍA

El presente estudio es de clase pre-experimental con un grupo experimental: El diseño más habitual de este tipo de investigación es el estudio antes-después (o pre-post) de un sólo grupo o con grupo de control no equivalente. Este tipo de diseño se basa en la medición y comparación de la variable respuesta antes y después de la exposición del sujeto a la intervención experimental. Se compararon los resultados para el grupo con pre y el pos-test permitiendo demostrar la hipótesis del trabajo y la efectividad del software para mejorar la calidad, a través del uso del software los programas desarrollados en RPG incrementaron sus calificaciones para los índices de performance, mantenibilidad, buenas prácticas, malas prácticas, complejidad y obsolescencia, como se evidencia en los resultados.

9. RESULTADOS

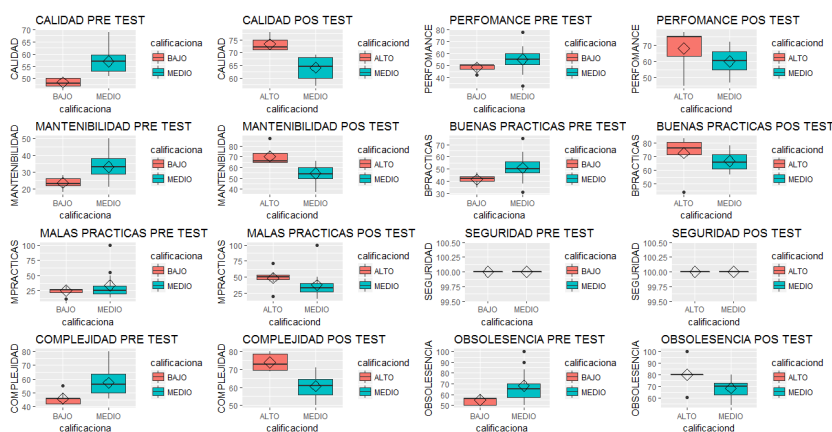


Figura 2. Resultados pre y pos test.

En la figura 2 se muestra los boxplots de los datos para la variable calidad y los diversos indicadores, se puede concluir que se ha mejorado ya que los valores con calificación bajo no existen y se ve valores mayores para calificaciones alto y medio.

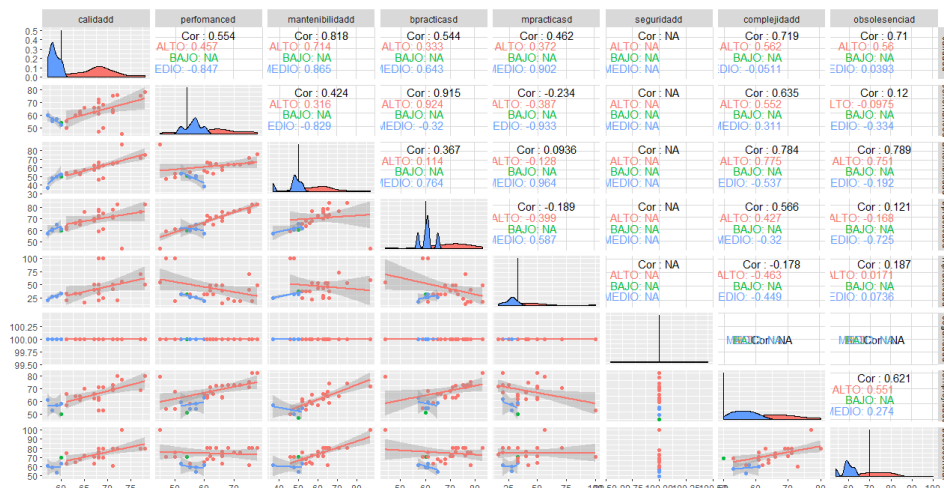


Figura 3. Matriz de correlacion.

La figura 3 muestra un graficoscatterplotmultiple para la variable calidad y sus indicadores mantenibilidad, buenas practicas, malas practicas, seguridad, complejidad y obsolescencia. La calidad tiene una fuerte correlacion con los indicadores performance, mantenibilidad, buenas prácticas, malas prácticas, complejidad y obsolescencia observándose una relación positiva y una baja correlacion se presenta entre las dimensiones performance, mantenibilidad y buenas prácticas con malas prácticas observándose una relación negativa. Complejidad y malas prácticas también presenta una relación negativa. Performance, malas prácticas y buenas prácticas con obsolescencia presentan una

relación negativa.

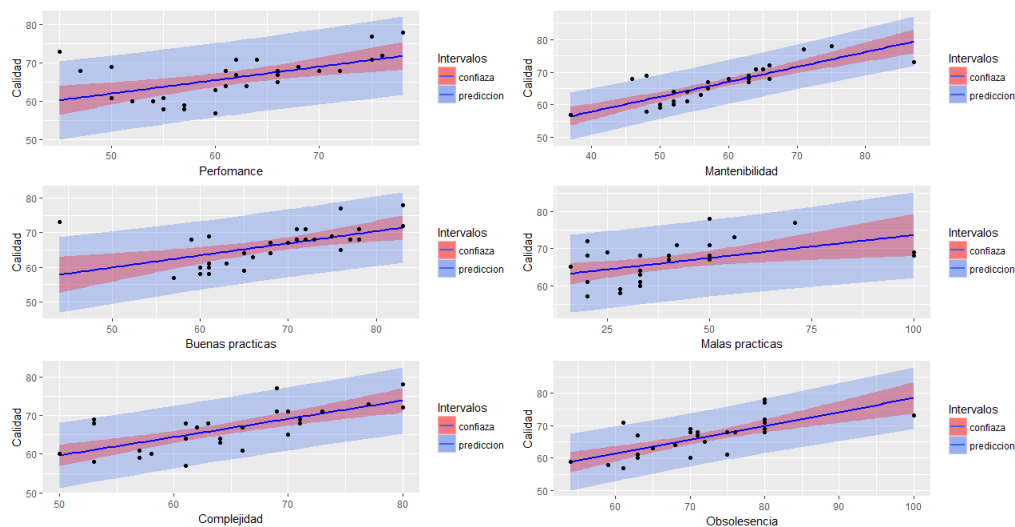


Figura 4. Intervalos de confianza.

La figura 4 muestra los intervalos de confianza (rojo) o posibles valores basados en la evidencia dentro de los cuales puede estar el valor de la media de la población y los intervalos de predicción (azul) donde se ubicarían los valores predichos, para la regresión calidad – indicadores. El intervalo de predicción sugiere valores mayores aun cercanos a la recta, el 95% de las nuevas observaciones estarán contenidas en esta banda, apoyando el supuesto de que el uso de la herramienta mejora la calidad de las aplicaciones. La banda de predicción (exterior) garantiza que el 95% de las nuevas observaciones estarán contenidas en esta banda.

El intervalo de confianza garantiza que la media de la población caerá dentro de esta banda.

DECLARACIÓN JURADA DE AUTORÍA Y AUTORIZACIÓN PARA LA PUBLICACIÓN DEL ARTÍCULO CIENTÍFICO

Yo, Giovanni Barrero Ortiz, estudiante (X), egresado (), docente (), del Programa de Maestría en Gestión de Tecnología de Información de la Escuela de Postgrado de la Universidad César Vallejo, identificado(a) con DNI N° 48839223, con el artículo titulado: "Sistema QSOURCE en la calidad del software desarrollado en RPG" Declaro bajo juramento que:

- 1) El artículo pertenece a mi autoría.
- 2) El artículo no ha sido plagiada ni total ni parcialmente.
- 3) El artículo no ha sido autoplagiada; es decir, no ha sido publicada ni presentada anteriormente para alguna revista.
- 4) De identificarse la falta de fraude (datos falsos), plagio (información sin citar a autores), autoplagio (presentar como nuevo algún trabajo de investigación propio que ya ha sido publicado), piratería (uso ilegal de información ajena) o falsificación (representar falsamente las ideas de otros), asumo las consecuencias y sanciones que de mi acción se deriven, sometiéndome a la normatividad vigente de la Universidad César Vallejo.
- 5) Si, el artículo fuese aprobado para su publicación en la Revista u otro documento de difusión, cedo mis derechos patrimoniales y autorizo a la Escuela de Postgrado, de la Universidad César Vallejo, la publicación y divulgación del documento en las condiciones, procedimientos y medios que disponga la Universidad.

Lugar y fecha: Los Olivos Lima, 18 de marzo del 2018 Giovanni Barrero Ortiz DNI
N° 48839223

Anexo 2. Matriz de consistencia

ANEXO 1: MATRIZ DE CONSISTENCIA						
TÍTULO: SISTEMA QSOURCE EN LA MEJORA DE LA CALIDAD DEL SOFTWARE (RPG)						
AUTOR: GIOVANNI BARRERO ORTIZ.						
PROBLEMA	OBJETIVOS	HIPÓTESIS	VARIABLES E INDICADORES			
<p>PROBLEMA PRINCIPAL: ¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora de la calidad de Software hecho en RPG en una Institución Bancaria?</p> <p>PROBLEMAS SECUNDARIOS</p> <p>¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora del performance del software hecho en RPG de Software hecho en RPG en una Institución Bancaria?</p> <p>¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora de la mantenibilidad del software hecho en RPG de Software hecho en RPG en una Institución Bancaria?</p> <p>¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora de buenas prácticas en el software hecho en RPG en una Institución Bancaria?</p> <p>¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora de malas prácticas de codificación en el software hecho en RPG en una Institución Bancaria?</p>	<p>OBJETIVO GENERAL Determinar cuales el efecto de la implementación del Sistema QSOURCE en la mejora de la calidad de Software hecho en RPG en una Institución Bancaria</p> <p>OBJETIVOS ESPECÍFICOS</p> <p>Determinar el efecto de la implementación del Sistema QSOURCE en la mejora del performance del software hecho en RPG de Software hecho en RPG en una Institución Bancaria</p> <p>Determinar el efecto de la implementación del Sistema QSOURCE en la mejora de la mantenibilidad del software hecho en RPG en una Institución Bancaria</p> <p>Determinar el efecto de la implementación del Sistema QSOURCE en la mejora de buenas practicas de Software hecho en RPG en una Institución Bancaria</p> <p>Determinar el efecto de la implementación del Sistema QSOURCE en la mejora de malas prácticas de codificación en el software hecho en RPG en una Institución Bancaria</p>	<p>HIPÓTESIS GENERAL El sistema QSOURCE mejora de la calidad de Software hecho en RPG en una Institución Bancaria</p> <p>HIPÓTESIS ESPECÍFICAS</p> <p>El sistema QSOURCE mejora el performance del software hecho en RPG en una Institución Bancaria</p> <p>El sistema QSOURCE mejora la mantenibilidad del software hecho en RPG en una Institución Bancaria</p> <p>El sistema QSOURCE mejora las buenas practicas en el software hecho en RPG en una Institución Bancaria</p> <p>¿El sistema QSOURCE mejora las malas prácticas de codificación en el software hecho en RPG en una Institución Bancaria?</p>	<p>Variable Independiente: SISTEMA QSOURCE</p>			
			Propuesta del Sistema			
			<p>Variable Dependiente: CALIDAD DEL SOFTWARE (RPG)</p>			
			Dimensiones	Indicadores	Ítems	Niveles y rangos
			Complejidad	QC2LE , BNDDIR , DISK , OPTIMIZE , EXTPGM , IMPORT CALL , CALLB , EXTPROC , CALLP , GOTO , EXSR , IF , DOW , DOU , FOR , QCMDEXC SELECT * , /COPY , ALLOC BITOFF , BITON , LINEAS FUENTE , LINEAS HOJAS F LINEAS HOJAS I , LIEAS HJAS E LINEAS HOJA D , LINEAS HOJA D LINEAS HOJA C , LINEAS HOJA P LINEAS HOJA O , HOJA FREE LINEAS VAR.POINTER TOTAL DE IMPORT , RUNSQLSTM , OPNQRYF LINEAS X RUTINA LINEAS X PROGRAMA NIVEL ANIDAMIENTO IF1 NIVEL ANIDAMIENTO DOW NIVEL ANIDAMIENTO DOU NIVEL ANIDAMIENTO FOR		Bajo = (01-40) Medio = (41-51) Alto = (62-100)

<p>¿Cuál es el efecto de la implementación del Sistema QSOURCE en la seguridad del software hecho en RPG en una Institución Bancaria?</p> <p>¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora de la complejidad del software hecho en RPG de Software hecho en RPG en una Institución Bancaria?</p> <p>¿Cuál es el efecto de la implementación del Sistema QSOURCE en la mejora de la mantenibilidad del software hecho en RPG de Software hecho en RPG en una Institución Bancaria?</p>	<p>Determinarel efecto de la implementación del Sistema QSOURCE en la seguridad del software hecho en RPG en una Institución Bancaria</p> <p>Determinarel efecto de la implementación del Sistema QSOURCE en la mejora de la complejidad del software hecho en RPG en una Institución Bancaria</p> <p>Determinarel efecto de la implementación del Sistema QSOURCE en la mejora de la mantenibilidad del software hecho en RPG en una Institución Bancaria</p>	<p>El sistema QSOURCE mejora la seguridad en el software hecho en RPG en una Institución Bancaria</p> <p>El sistema QSOURCE mejora la complejidad del software hecho en RPG en una Institución Bancaria</p> <p>El sistema QSOURCE mejora la mantenibilidad del software hecho en RPG en una Institución Bancaria</p>	<p>Seguridad</p> <p>Obsolescence</p> <p>BuenasPrácticas</p>	<p>GRTOBJAUT TOTAL PROCEDIMIENTOS TOTAL VARIABLES INDEP. TOTAL MODULOS ILE TOTAL *SRVPGM</p> <p>qcmd , strqm, strdfu, strpdm, chkobj, grtobjaut</p> <p>goto , add, cat, div, mult mvr , sqrt, z-add , z-sub , cab mhlzo, mhhzo, lineas hojas i, lineas hoja e, lineas hoja o</p> <p>qc2le, bddir, actgrp, optimize, extpgm, import call, callb, extproc, callp goto, exsr, if ,dow, dou, for qcmdexc, select *, /copy %date, %time, %timestamp %diff, %subdt, %day, %hours, %miutes, %months %months, %seconds, %years %char, %dec, chain, reade readpe, setll, setgt, %fields, programa, objetivo requerimiento, fecha memo fecha atencion, analista modificado por modificaciones maxlin fuentes *in lineas hoja f lineas hoja i lineas hoja e lineas hoja d lineas hojas c linneas hojas p lineas hoja o lineas hoja h lineas hoja free lineasvar.pointer var. no usadas rutinas no usadas cod.quemado tot. export tot.import cpyf, runqry, crtpf, ovrdbf rclstg, runsqlstm, opnqryf strqm, chkobj, tot.linrutina</p>		
--	--	--	---	--	--	--

				tot. lin. por procedimto inz anidamiento if anidamientodou	
			MalasPrácticas	qcmd, usrop, disk, printer extpgm, close, add, cat div	
			Performance	qcmd, bnmdir, actgrp optimize, usropn, disk printer, const, extpgm import, call, callb, goto exsr, if, dow, dou, for, qcmd runqry, crt, crt, ovrdbrf rclstg, runsqlstm, opnqryf strqm, strdfu, strpdm import , call , callb , extproc , exsr , if , dow dou , for , qcmdexec , open , close , select * /copy , %date , %time , %timestamp , %diff	
			Mantenibilidad	%subdt , %day , %hours , %minutes , %month %mseconds , %second , %years , %char , %dec %fields , programa , objetivo , requerimiento , fecha memo fecha atención , analista , modificaciones , free , líneas hojas f , lineas hojas d , líneas	

				hojas c , líneas hojas h , líneas hoja free , líneas def. var. poi , total variables no usadas total rutinas no usadas , total líneas código quemado total export , total import , total líneas rutina , Total lineaprocedimiento , uso de inz() , nivel anida. if Nivel anida. do , nivel anida. dou , nivel anida. dow Nivel anida. for , total procedimientos , total variables i total modulosile , total modulos *srvgm		
--	--	--	--	---	--	--

TIPO Y DISEÑO DE INVESTIGACIÓN	POBLACIÓN Y MUESTRA	TÉCNICAS E INSTRUMENTOS	ESTADÍSTICA DESCRIPTIVA E INFERENCIAL			

<p>TIPO: APLICATIVO El tipo de investigación es aplicada, al respecto Murillo (2008), refiere que: la investigación aplicada recibe el nombre de “investigación práctica o empírica”, que se caracteriza porque busca la aplicación o utilización de los conocimientos adquiridos, a la vez que se adquieren otros, después de implementar y sistematizar la práctica basada en investigación. El uso del conocimiento y los resultados de investigación que da como resultado una forma rigurosa, organizada y sistemática de conocer la realidad.</p> <p>NIVEL: Experimental. DISEÑO: EXPERIMENTAL Esta investigación corresponde al diseño experimental debido a que “los diseños experimentales se utilizan cuando el investigador pretende establecer el posible efecto de una causa que se manipula” (Hernández y otros, 2010, p. 122). Es de clase pre-experimental, los sujetos de la muestra de estudio fueron asignados de forma intencional.</p>	<p>POBLACIÓN: -</p> <p>TAMAÑO DE MUESTRA: 30 programas</p> <p>TIPO DE MUESTRA: No probabilístico</p>	<p>Variable Independiente: SISTEMA QSOURCE Instrumentos: Propuesta Software Autor: BARRERO. Año: 2017 Ámbito de Aplicación: Empresa bancaria Forma de Administración: Directa</p> <p>Variable Proceso de Selección Técnicas: Fichaje Instrumentos: Reporte Estadístico de Calidad del Software Autor: BARRERO. Año: 2017 Ámbito de Aplicación: Empresa Bancaria Forma de Administración: Directa</p>	<p>DESCRIPTIVA: De distribución de frecuencia, tablas de contingencia, figuras</p> <p>DE PRUEBA: Prueba hipótesis</p> <p>Para Torres (2007) “La hipótesis es un planteamiento que establece una relación entre dos o más variables para explicar y, si es posible, predecir probabilísticamente las propiedades y conexiones internas de los fenómenos o las causas y consecuencias de un determinado problema” (p. 129)</p>
---	---	---	--

Anexo 3

Validación del instrumento



CERTIFICADO DE VALIDEZ DE CONTENIDO DEL INSTRUMENTO QUE MIDE

Nº	DIMENSIONES / ítems	Pertinencia ¹		Relevancia ²		Claridad ³		Sugerencias
		Si	No	Si	No	Si	No	
DIMENSION 1. Performance								
1	Instrucciones: qe2le, brddir, disk, optimizar, extpgm, import, pto/hojas	X		X		X		
2	Instrucciones: call, callb, extproc, callp, goto, exar, if, dow, dos, for, qendvoc	Y		X		X		
3	Instrucciones: select *, copy, alloc, hnsff, hnos	X		X		Y		
4	líneas hojas f, hojas i, hojas e, hoja d, hoja c, hoja p, hoja o, hoja free.	X		X		Y		
5	Líneas var, pointer	X		X		Y		
6	total, de import, ruroqbitm, qnqvprf, líneas fuente	X		X		Y		
7	líneas x rutinas, x programa	X		X		X		
8	nivel anidamiento if, dow, dos, for	X		X		X		
9	total procedimientos, variables indep., módulos ile, *srvgpm	X		X		Y		
DIMENSION 2. Mantenibilidad								
1	Instrucciones: import, call, callb, extproc, exar, if, dow, dos, for, qendvoc, open, close, select *	Y		X		Y		
2	copy, %date, %time, %timeamp, %diff	Y		X		Y		
3	Funciones %cccc del RPG	X		X		X		
4	Programa, objetivo, requerimiento, fecha memo	X		X		X		
5	fecha atención, analista, modificaciones, free, líneas hojas f, líneas hojas d, líneas hojas c, líneas hojas h, líneas hoja free, líneas def. var. poi, total variables no usadas	X		X		X		
6	total rutinas no usadas, total líneas código quemado	X		X		Y		
7	total export, total import, total líneas rutina,	X		X		X		
8	Total, línea procedimiento, uso de ins (i), nivel anida, if	X		X		X		
9	Nivel anida, do, nivel anida, Dos, nivel anida, dow	X		X		X		
10	Nivel anida, for, total procedimientos, total variables i	X		X		X		
11	total módulos ile, total módulos *srvgpm	X		X		X		
12	Instrucciones: import, call, callb, extproc, exar, if, dow	X		X		X		
13	dos, for, qendvoc, open, close, select *	X		X		X		
DIMENSION 3. Buenas prácticas.								
1	qe2le, brddir, actgpp, optimizar, extpgm, import, call, callb, extproc, callp, goto, exar, if, dow, dos, for, qendvoc, select *, copy, *in, ins	Y		X		Y		
2	Funciones %cccc	X		X		X		
3	maxlpe, setll, setpl, %fields, programa, objetivo	X		X		X		
4	requerimiento, fecha memo, fecha atención, analista, modificado por, modificaciones	X		Y		X		
5	maxlín fuentes	X		Y		X		
6	líneas hoja f, hoja i, hoja e, hoja d, hojas c, hojas p, hoja o, hoja h	X		X		X		
7	hoja free, var, pointer	X		X		X		
8	Var. no usadas, rutinas no usadas	X		X		X		

9	cod.quemado	X		X		X		
10	tot. export, tot. Import	X		X		X		
11	Comandos: cpxf, runqy, crtpf, ovrdbf, rclstg, runsqstm, opnqyf	X		X		X		
12	strqm, chkobj, tot.linrutina	X		X		X		
13	tot. fin. por procedimiento	X		X		X		
14	anidamiento if	X		X		X		
15	anidamiento dou	X		X		X		
16	qc2le, bnddir, actgrp, optimize, extpgm, import, call, callb, extproc, callp, goto, exsr, if, dow, dou, for, qcmdexe, select *, /copy, *m, inz	X		X		X		
17	Funciones %XXXX	X		X		X		
DIMENSION 4. Malas prácticas.								
1	Instrucciones: qcmd, usrop, disk, printer, extpgm, close, add, cat div, goto	X		X		X		
DIMENSION 5. Seguridad.								
1	Comandos: qcmd, strqm, strdfa, strpdm, chkobj, grtobjaut	X		X		X		
DIMENSION 6. Complejidad								
1	Instrucciones: qc2le, bnddir, disk, optimize, extpgm, import, call, callb, extproc, callp, goto, exsr, if, dow, dou, for, qcmdexe, select *, /copy, alloc, bisoff, bkon,	X		X		X		
2	líneas fuente, líneas hojas f, líneas hojas i, hojas e, hoja d, hoja d, hoja c, hoja p, hoja o, hoja free, líneas var.pointer	X		X		X		
3	total de import, runsqstm, opnqyf	X		X		X		
4	líneas x rutina, x programa	X		X		X		
5	nivel anidamiento if, anidamiento dow, anidamiento dou, nivel anidamiento for	X		X		X		
6	grtobjaut	X		X		X		
7	total: procedimientos, variables indep, módulos ile	X		X		X		
8	total *srpgm	X		X		X		
9	qc2le, bnddir, disk, optimize, extpgm, import	X		X		X		
DIMENSION 7. Obsolescencia								
1	Comandos: goto, add, cat, div, mult mvr, sqrt, z-add, z-sub, cab, mhizo, mhizo, líneas hojas i, líneas hoja c, líneas hoja o	X		X		X		

Observaciones (precisar si hay suficiencia): Si hay suficiencia

Opinión de aplicabilidad: Aplicable Aplicable después de corregir [] No aplicable []

Apellidos y nombres del juez validador: Dr/ Mg Flores Sotelo Willean DNI: 06175729

Especialidad del validador: Guatemala Escania expert / Escania

9 de 12 del 2016
 Dr. Willean Sebastian Flores Sotelo
 Docente-Investigador de Posgrado
 CEL N° 09426
 Firma del Experto Informante.



CERTIFICADO DE VALIDEZ DE CONTENIDO DEL INSTRUMENTO QUE MIDE

N°	DIMENSIONES / Items	Pertinencia ¹		Relevancia ²		Claridad ³		Sugerencias
		Si	No	Si	No	Si	No	
DIMENSION 1. Performance								
1	Instrucciones: qc2le, bnddir, disk, optimize, extpgm, import, grtobjaut	X		X		X		
2	Instrucciones: call, callb, extproc, callp, goto, exsr, if, dow, dou, for, qcmdexe	X		X		X		
3	Instrucciones: select *, /copy, alloc, bisoff, bkon,	X		X		X		
4	líneas hojas f, hojas i, hojas e, hoja d, hoja c, hoja p, hoja o, hoja free.	X		X		X		
5	Líneas var. pointer	X		X		X		
6	total de import, runsqstm, opnqyf, líneas fuente	X		X		X		
7	líneas x rutina, x programa	X		X		X		
8	nivel anidamiento if, dow, dou, for	X		X		X		
9	total procedimientos, variables indep, módulos ile, *srpgm	X		X		X		
DIMENSION 2. Mantabilidad.								
1	Instrucciones: import, call, callb, extproc, exsr, if, dow, dou, for, qcmdexe, open, close, select *	X		X		X		
2	/copy, %date, %time, %timestamp, %skid	X		X		X		
3	Funciones %XXXX del RPG	X		X		X		
4	Programa, objetivo, requerimiento, fecha memo	X		X		X		
5	fecha creación, analista, modificaciones, free, líneas hojas f, líneas hojas d, líneas hojas c, líneas hojas h, líneas hoja free, líneas def. var. pos, total variables no usadas	X		X		X		
6	total rutinas no usadas, total líneas código quemado	X		X		X		
7	total export, total import, total líneas rutina,	X		X		X		
8	Total línea procedimiento, uso de ise (), nivel anida, if	X		X		X		
9	Nivel anida. do, nivel anida. Dou, nivel anida. dow	X		X		X		
10	Nivel anida. for, total procedimientos, total variables i	X		X		X		
11	total módulos ile, total módulos *srpgm	X		X		X		
12	Instrucciones: import, call, callb, extproc, exsr, if, dow	X		X		X		
13	dou, for, qcmdexe, open, close, select *	X		X		X		
DIMENSION 3. Buenas prácticas.								
1	qc2le, bnddir, actgrp, optimize, extpgm, import, call, callb, extproc, callp, goto, exsr, if, dow, dou, for, qcmdexe, select *, /copy, *m, inz	X		X		X		
2	Funciones %XXXX	X		X		X		
3	realpr, stll, srgr, %fields, programa, objetivo	X		X		X		
4	requerimiento, fecha memo, fecha creación, analista, modificado por, modificaciones	X		X		X		
5	maxlin fuentes	X		X		X		
6	líneas hoja f, hoja i, hoja e, hoja d, hojas c, hojas p, hoja o, hoja h	X		X		X		
7	hoja free, var. pointer	X		X		X		
8	Var. no usadas, rutinas no usadas	X		X		X		

9	cod.quemado	x		x		x			
10	tot. export. tot. import	x		x		x			
11	Comandos: copy, runqy, cript, ovrdbf, relstg, runsqlstm, opnqyvf	x		x		x			
12	strqm, chkobj, tot.linrutina	x		x		x			
13	tot. lin. por procedimiento	x		x		x			
14	anidamiento if	x		x		x			
15	anidamiento dou	x		x		x			
16	qc2le, bnddir, setgrp, optimize, extpgm, import, call, callb, exproc, callp goto, exsr, if, dow, dou, for, qcmdexe, select *, /copy, *in, inz	x		x		x			
17	Funciones %SXXX	x		x		x			
DIMENSIÓN 4. Malas prácticas.									
1	Instrucciones: qcmd, usrop, disk, printer, extpgm, close, add, cat div, goto	SI	No	SI	No	SI	No		
DIMENSIÓN 5. Seguridad.									
1	Comandos: qcmd, strqm, strdfu, strpdm, chkobj, gtrobjaut	SI	No	SI	No	SI	No		
DIMENSIÓN 6. Complejidad									
1	Instrucciones: qc2le, bnddir, disk, optimize, extpgm, import, call, callb, exproc, callp, goto, exsr, if, dow, dou, for, qcmdexe, select *, /copy, alloc, biofff, biton,	x		x		x			
2	líneas fuente, líneas hojas f, líneas hojas i, hojas e, hoja d, hoja d, hoja c, hoja p, hoja o, hoja free, líneas var.pointer	x		x		x			
3	total de import, runsqlstm, opnqyvf	x		x		x			
4	líneas x rutina, x programa	x		x		x			
5	nivel anidamiento if, anidamiento dow, anidamiento dou, nivel anidamiento for	x		x		x			
6	gtrobjaut	x		x		x			
7	total: procedimientos, variables indep, módulos ile	x		x		x			
8	total *srvgpm	x		x		x			
9	qc2le, bnddir, disk, optimize, extpgm, import	x		x		x			
DIMENSIÓN 7. Obsolescencia									
1	Comandos: goto, add, cat, div, mult mvr, sqrt, z-add, z-sub, cab, mhlzo, mhlzo, líneas hojas i, líneas hoja c, líneas hoja o	SI	No	SI	No	SI	No		

Observaciones (precisar si hay suficiencia): _____

Opinión de aplicabilidad: **Aplicable [x]** **Aplicable después de corregir []** **No aplicable []**

Apellidos y nombres del juez validador: Dr/ Mg: Eliana Castro de Rosalynn DNI: 40650095

Especialidad del validador: Sesión de Tecnologías de la Información / Administración de la Educación

4 de 12 del 2016

- *Pertinencia: El ítem corresponde al concepto teórico formulado.
- *Relevancia: El ítem es apropiado para representar al componente o dimensión específica del constructo
- *Claridad: Se entiende sin dificultad alguna el enunciado del ítem, es conciso, exacto y directo

Nota: Suficiencia, se dice suficiencia cuando los ítems planteados



 Firma del Experto Informante.



CERTIFICADO DE VALIDEZ DE CONTENIDO DEL INSTRUMENTO QUE MIDE

Nº	DIMENSIONES / Ítems	Pertinencia ¹		Relevancia ²		Claridad ³		Sugerencias
		SI	No	SI	No	SI	No	
DIMENSIÓN 1. Performance								
1	Instrucciones: qc2le, bnddir, disk, optimize, extpgm, import, gtrobjaut	x		x		x		
2	Instrucciones: call, callb, exproc, callp, goto, exsr, if, dow, dou, for, qcmdexe	x		x		x		
3	Instrucciones: select *, /copy, alloc, biofff, biton,	x		x		x		
4	líneas hoja f, hojas i, hojas e, hoja d, hoja c, hoja p, hoja o, hoja free,	x		x		x		
5	Líneas var. pointer	x		x		x		
6	total de import, runsqlstm, opnqyvf, líneas fuente	x		x		x		
7	líneas x rutina, x programa	x		x		x		
8	nivel anidamiento if, dow, dou, for	x		x		x		
9	total procedimientos, variables indep, módulos ile, *srvgpm	x		x		x		
DIMENSIÓN 2. Mantabilidad.								
1	Instrucciones: import, call, callb, exproc, exsr, if, dow, dou, for, qcmdexe, open, close, select *	x		x		x		
2	/copy, %date, %time, %timestamp, %chrf	x		x		x		
3	Funciones %SXXXX del RPG	x		x		x		
4	Programa, objetivo, requerimiento, fecha mesa	x		x		x		
5	fecha atendida, analista, modificaciones, frea, líneas hojas f, líneas hojas d, líneas hojas o, líneas hojas h, líneas hoja free, líneas def. var. poi, total variables no usadas	x		x		x		
6	total rutinas no usadas, total líneas código quemado	x		x		x		
7	total export, total import, total líneas rutina,	x		x		x		
8	Total, línea procedimiento, uso de inz (), nivel anida, if	x		x		x		
9	Nivel anida, do, nivel anida, Dou, nivel anida, dow	x		x		x		
10	Nivel anida, for, total procedimientos, total variables i	x		x		x		
11	total módulos ile, total módulos *srvgpm	x		x		x		
12	Instrucciones: import, call, callb, exproc, exsr, if, dow	x		x		x		
13	dou, for, qcmdexe, open, close, select *	x		x		x		
DIMENSIÓN 3. Buenas prácticas.								
1	qc2le, bnddir, setgrp, optimize, extpgm, import, call, callb, exproc, callp goto, exsr, if, dow, dou, for, qcmdexe, select *, /copy, *in, inz	x		x		x		
2	Funciones %SXXXX	x		x		x		
3	readpe, outll, senq, %fields, programa, objetivo	x		x		x		
4	requerimiento, fecha mesa, fecha atendida, analista, modificado por, modificaciones	x		x		x		
5	maxlin fuentes	x		x		x		
6	líneas hoja f, hoja l, hoja e, hoja d, hojas c, hojas p, hoja o, hoja h	x		x		x		
7	hoja free, var. pointer	x		x		x		
8	Var. no usadas, rutinas no usadas	x		x		x		

GOTO , EXSR , IF , DOW , DOU , FOR , QCMDEXC , OPEN , CLOSE , SELECT , CHAIN , READ , READE , READPE , SETLL , SETGT , *IN , NUMERO DE LINEAS DE HOJAS , CPYF , RUNQRY , CRTLF , CTRPF , OVRDBF , RCLSTG , RUNSQLSTM , OPNQRYF , STRQM , STRDFU , STRPDM , CHKOBJ ,NIVEL DE ANIDAMIENTTO DE SENTENCIAS DE CONTROL.									
---	--	--	--	--	--	--	--	--	--

Observaciones (precisar si hay suficiencia):

Si hay suficiencia

Opinión de aplicabilidad: Aplicable

Aplicable después de corregir

No aplicable

4 de Diciembre del 2016

Apellidos y nombres del juez evaluador: DR. JORGE RAFAEL DIAZ DUMONT DNI 08698815

Especialidad del evaluador: POST DOCOTARO EN ADMINISTRACIÓN PÚBLICA - ING. INDUSTRIAL.

¹Claridad: Se entiende sin dificultad alguna el enunciado del ítem, es conciso, exacto y directo

²Pertinencia: Si el ítem pertenece a la dimensión.

³Relevancia: El ítem es apropiado para representar al componente o dimensión específica del constructo

Nota: Suficiencia, se dice suficiencia cuando los ítems planteados son suficientes para medir la dimensión

Anexo 4 Programa de aplicación

```

SOU0000003          QSOURCE RPG CODE QUALITY ANALIZER          8/01/16
SISTEMA2            MIEMBROS LIBRERÍA: OBARRERO FUENTE : PROPUESTAS 10:03:44
-----
02=Modificar,03=Copiar,04=Eliminar,05=Consultar,06=Analizar interactivo
07=Analizar Batch 08=Estad. 09=Editar 10=Var.no u. 11=Hard code

OP  MIEMBRO      TIPO      DESCRIPCION
06  DL6334       RPGLE      Ingreso de Reenganche
___ DP0112        SQLRPGLE   CONSULTA TARJ/PLD
___ DP0114        RPGLE      SIMULADOR PROPUESTAS CONVENIO
___ DP0115        RPGLE      Mantenimiento Propuestas de Cr
___ DP0124        RPGLE      MANTENIMIENTO DE PROPUESTAS
___ DP0150        SQLRPGLE   LIQUIDEZ DEL SECTOR PUBLICO
___ DP0155        SQLRPGLE   REPORTE CREDITICIO CONSOLIDADO
___ DP0156        SQLRPGLE   REPORTE CREDITICIO CONSOLIDADO
___ DP0158        RPGLE      CONSULTA DE CLIENTES
___ DP0159        RPGLE      CONSULTA DE CLIENTES

F3=Salir F6=Crear F7=Análisis Masivo F8=reporte F9=Estad.
RollUp=AvanReg RollDw=RetrReg

```

Figura 86. Analizar programa interactivamente.

```

SOU0000006          QSOURCE RPG CODE QUALITY ANALIZER          /16
SISTEMA2            ESTADISTICAS DL6334                        :32
-----
05=Consultar 09=Reporte

OP  FECHA      HORA      PERFOMA CALIDAD BEST.PR BAD.PRA SEGURID COMPLEJ OBSOLE
___ 2016-01-08 10041060    59      50      58      33      55

F1=Ayuda F3=Salir F6=Crear
RollUp=AvanReg RollDw=RetrReg

```

Figura 87. Estadísticas programa.

SOU0000008		SISTEMA2		SOUR		OSOURCE RESUMEN ESTADISTICA DL6334				01/16		
Perfomance	19 /	69	Calidad	17 /	86							
Best Pract.	18 /	86	Bad Pract.	4 /	24							
Seguridad	/	6	Complejidad	11 /	45							
Obsolesenc.	/	18										
OP	METRICA	VALOR	PARAMETROS				VLR.INI.	VLR.FINAL				
—	IMPORT		S	S	S	N	N	S	N	1	100	1
—	CALL	27	S	S	S	N	N	S	N	1	20	1
—	CALLB		S	S	S	N	N	S	N	1	20	1
—	EXTPROC		S	S	S	N	N	S	N	1	20	0
—	CALLP	1	S	S	S	N	N	S	N	1	20	0
—	GOTO		S	S	S	N	N	S	S			0
—	EXSR	107	S	S	S	N	N	S	N	1	50	1
—	IF	367	S	S	S	N	N	S	N	1	10	1
—	DOW	23	S	S	S	N	N	S	N	1	10	1
—	DOU		S	S	S	N	N	S	N	1	10	0
F3=Salir												

Figura 90. Resumen de estadísticas de programa 1.

SOU0000008		SISTEMA2		SOUR		OSOURCE RESUMEN ESTADISTICA DL6334				01/16		
Perfomance	19 /	69	Calidad	17 /	86							
Best Pract.	18 /	86	Bad Pract.	4 /	24							
Seguridad	/	6	Complejidad	11 /	45							
Obsolesenc.	/	18										
OP	METRICA	VALOR	PARAMETROS				VLR.INI.	VLR.FINAL				
—	FOR		S	S	S	N	N	S	N	1	10	0
—	QCMDEXC		S	S	S	N	N	S	N	1	10	0
—	OPEN	1	S	S	N	S	N	N	N	1	10	0
—	CLOSE	1	S	S	N	S	N	N	N	1	10	0
—	SELECT *		S	S	S	N	N	S	N	1	20	1
—	/COPY		N	S	S	N	N	S	N	1	10	0
—	ADD	258	N	S	N	S	N	N	S			1
—	CAT		N	S	N	S	N	N	S			0
—	DIV	9	N	S	N	S	N	N	S			1
—	MULT	14	N	S	N	S	N	N	S			1
F3=Salir												

Figura 91. Resumen de estadísticas de programa 2.

SOU0000008		SISTEMA2		SOU		OSOURCE RESUMEN ESTADISTICA DL6334				01/16	
Perfomance	19 /	69	Calidad	17 /	86						
Best Pract.	18 /	86	Bad Pract.	4 /	24						
Seguridad	/	6	Complejidad	11 /	45						
Obsolesenc.	/	18									
OP	METRICA	VALOR	PARAMETROS				VLR.INI.	VLR.FINAL			
—	MLHZO		N	S	N	S	N	N	S	0	
—	%DATE		N	S	S	N	N	N	N	1	999 0
—	%TIME		N	S	S	N	N	N	N	1	999 00
—	%TIMESTAMP		N	S	S	N	N	N	N	1	999 000
—	%DIFF		N	S	S	N	N	N	N	1	999 000
—	%SUBDT		N	S	S	N	N	N	N	1	999 000
—	%DAY		N	S	S	N	N	N	N	1	999 000
—	%HOURS		N	S	S	N	N	N	N	1	999 000
—	%MINUTES		N	S	S	N	N	N	N	1	999 000
—	%MONTHS		N	S	S	N	N	N	N	1	999 0
F3=Salir											

Figura 92. Resumen de estadísticas de programa 4.

SOU0000008		SISTEMA2		SOL		OSOURCE RESUMEN ESTADISTICA DL6334				3/01/16	
Perfomance	19 /	69	Calidad	17 /	86						
Best Pract.	18 /	86	Bad Pract.	4 /	24						
Seguridad	/	6	Complejidad	11 /	45						
Obsolesenc.	/	18									
OP	METRICA	VALOR	PARAMETROS				VLR.INI.	VLR.FINAL			
—	%MSECONDS		N	S	S	N	N	N	N	1	999 0
—	%SECONDS		N	S	S	N	N	N	N	1	999 0
—	%YEARS		N	S	S	N	N	N	N	1	999 00
—	%CHAR	6	N	S	S	N	N	N	N	1	999 000
—	%DEC		N	S	S	N	N	N	N	1	999 000
—	CHAIN	87	S	N	S	N	N	N	N	1	20 1
—	READE	28	S	N	S	N	N	N	N	1	20 1
—	READPE	2	S	N	S	N	N	N	N	1	20 00
—	SETLL	15	S	N	S	N	N	N	N	1	20 000
—	SETGT	3	S	N	S	N	N	N	N	1	20 0
F3=Salir											

Figura 93. Resumen de estadísticas de programa 5.

SOU0000008 SISTEMA2 SOUF		OSOURCE RESUMEN ESTADISTICA DL6334				'01/16					
Perfomance	19 /	69	Calidad	17 /	86						
Best Pract.	18 /	86	Bad Pract.	4 /	24						
Seguridad	/	6	Complejidad	11 /	45						
Obsolesenc.	/	18									
OP	METRICA	VALOR	.PARAMETROS.				VLR.INI.	VLR.FINAL			
—	%FIELDS		N	S	S	N	N	N	1	999	0
—	PROGRAMA	5	N	S	S	N	N	N	1	1	1
—	OBJETIVO	1	N	S	S	N	N	N	1	1	0
—	REQUERIMIENTO		N	S	S	N	N	N	1	1	1
—	FECHA MEMO		N	S	S	N	N	N	1	1	1
—	FECHA ATENCION		N	S	S	N	N	N	1	1	1
—	ANALISTA		N	S	S	N	N	N	1	1	1
—	MODIFICACIONES		N	S	S	N	N	N	1	1	1
—	LOS FUENTES NO DEBEN	5651	S	N	S	N	N	S	1	5000	1
—	*IN	221	S	S	S	N	N	S	1	1	1

F3=Salir

Figura 94. Resumen de estadísticas de programa 6.

SOU0000008 SISTEMA2 SOUF		OSOURCE RESUMEN ESTADISTICA DL6334				'01/16					
Perfomance	19 /	69	Calidad	17 /	86						
Best Pract.	18 /	86	Bad Pract.	4 /	24						
Seguridad	/	6	Complejidad	11 /	45						
Obsolesenc.	/	18									
OP	METRICA	VALOR	.PARAMETROS.				VLR.INI.	VLR.FINAL			
—	FREE	16	N	S	S	N	N	N	1	999	0
—	LINEAS HOJA F	1	S	S	S	N	N	S	1	40	0
—	LINEAS HOJA I	81	S	S	S	N	N	S			1
—	LINEAS HOJA E		S	S	S	N	N	S			0
—	LINEAS HOJA D	452	S	S	S	N	N	S	1	999	0
—	LINEAS HOJA C	3655	S	S	S	N	N	S	1	9999	0
—	LINEAS HOJA P	1	S	S	S	N	N	S	1	9999	0
—	LINEAS HOJA O		S	S	S	N	N	S	1	1	0
—	LINEAS HOJA H		S	S	S	N	N	N	1	10	1
—	LINEAS HOJA FREE		S	S	S	N	N	S	1	9999	0

F3=Salir

Figura 95. Resumen de estadísticas de programa 7.

SOU0000008 SISTEMA2 SOUI		OSOURCE RESUMEN ESTADISTICA DL6334				/01/16					
Perfomance	19 /	69	Calidad	17 /	86						
Best Pract.	18 /	86	Bad Pract.	4 /	24						
Seguridad	/	6	Complejidad	11 /	45						
Obsolesenc.	/	18									
OP METRICA	VALOR	..PARAMETROS.		VLR.INI.	VLR.FINAL						
LINEAS DEF. VAR. POI		S	S	S	N	N	S	N	1	1	0
TOTAL VARIABLES NO U	3	S	S	S	N	N	N	N			1
TOTAL RUTINAS NO USA		S	S	S	N	N	N	N			0
TOT. LIN. COD. QUEMADO	598	S	S	S	N	N	N	N			1
TOTAL EXPORT		S	S	S	N	N	N	N	1	999	0
TOTAL IMPORT		S	S	S	N	N	S	N	1	999	1
CPYF		S	N	S	N	N	N	N	1	5	0
RUNQRY		S	N	S	N	N	N	N	1	5	0
CRTF		S	N	S	N	N	N	N	1	5	0
CRTF		S	N	S	N	N	N	N	1	5	0
F3=Salir											

Figura 96. Resumen de estadísticas de programa 8.

SOU0000008 SISTEMA2 SOI		OSOURCE RESUMEN ESTADISTICA DL6334				3/01/16					
Perfomance	19 /	69	Calidad	17 /	86						
Best Pract.	18 /	86	Bad Pract.	4 /	24						
Seguridad	/	6	Complejidad	11 /	45						
Obsolesenc.	/	18									
OP METRICA	VALOR	..PARAMETROS.		VLR.INI.	VLR.FINAL						
OVRDBF		S	N	S	N	N	N	N	1	5	0
RCLSTG		S	N	S	N	N	N	N	1	5	0
RUNSQLSTM		S	N	S	N	N	S	N	1	1	0
OPNQRYP		S	N	S	N	N	S	N	1	1	0
STRQM		S	N	S	N	S	N	N	1	1	0
STRDFU		S	N	S	N	S	N	N	1	1	0
STRPDM		S	N	S	N	S	N	N	1	1	0
CHKOBJ		S	N	S	N	S	N	N	1	1	0
TOTAL LINEAS POR RUT	1259	S	S	S	N	N	S	N	1	30	1
TOTAL LINEAS POR PRO		S	S	S	N	N	S	N	1	30	0
F3=Salir											

Figura 97. Resumen de estadísticas de programa 9.

SOU0000008 SISTEMA2 SOUI		OSOURCE RESUMEN ESTADISTICA DL6334				/01/16				
Perfomance	19 /	69	Calidad	17 /	86					
Best Pract.	18 /	86	Bad Pract.	4 /	24					
Seguridad	/	6	Complejidad	11 /	45					
Obsolesenc.	/	18								
OP	METRICA	VALOR	..PARAMETROS.			VLR.INI.	VLR.FINAL			
___	USO DE INZ()		S	S	S	N	N	N	0	
___	NIVEL ANIDA.IF	6	S	S	S	N	N	S	1	15
___	NIVEL ANIDA.DOU		S	S	S	N	N	S	1	15
___	NIVEL ANIDA.DOW	1	S	S	S	N	N	S	1	15
___	NIVEL ANIDA.FOR		S	S	S	N	N	S	1	15
___	GRTOBJAUT		S	N	N	S	S	N		0
___	TOTAL DE PROCEDIMIEN	1	S	S	S	N	N	S	1	90
___	TOTAL DE VARIABLES I	320	N	S	N	S	N	S	1	1
___	TOTAL MODULOS ILE	1	S	S	S	N	N	S	1	10
___	TOTAL *SRVPGM	4	S	S	S	N	N	S	1	10

F3=Salir

Figura 98.Resumen de estadísticas de programa 10.

SOU0000006		QSOURCE CODE QUALITY ANALYZER				8/01/16			
SISTEMA2		ESTADISTICAS DL6334				10:08:30			
05=Consultar 09=Reporte									
OP	FECHA	HORA	PERFOMA	CALIDAD	BEST.PR	BAD.PRA	SEGRID	COMPLEJ	OBSOLE
___	2016-01-08	10041060	59	50	58	33		55	
09	2016-01-08	10045456	59	50	58	33		55	

F1=Ayuda F3=Salir F6=Crear
RollUp=AvanReg RollDw=RetrReg

Figura 99. Reportes de estadísticas.

SOU0000026
SISTEMA2

QSOURCE CODE QUALITY ANALYZER
ESTADISTICAS GENERALES DE CALIFICACION

28/01/16
16:34:07

05=Consultar

OP	PROGRAMA	FECHA	HORA	PERF	CAL	B.P	M.P	SEG	COP	OBS
—	DL6325	2016-01-28	16.32.08	60	45	57	22	00	73	00
—	DL6326	2016-01-28	16.32.08	60	45	57	22	00	73	00
—	DL6327	2016-01-28	16.32.08	71	50	66	40	00	78	00
—	DL6334	2016-01-28	16.32.08	59	50	58	33	00	55	00
—	DL6336	2016-01-28	16.32.08	63	50	68	20	00	64	00
—	DL6338	2016-01-28	16.32.08	80	72	78	80	00	83	00
—	DL6340	2016-01-28	16.32.08	69	50	66	44	00	71	00
—	DL6344	2016-01-28	16.32.08	72	52	72	25	00	75	00
—	DL6356	2016-01-28	16.32.08	68	55	70	25	00	76	00
—	DL6360	2016-01-28	16.32.08	80	66	80	57	00	84	00

F3=Salir
RollUp=AvanReg RollDw=RetrReg

Figura 100. Estadísticas generales.

SOU0000040

Consulta General de Calificación

Fecha.: 8/05/17
Usuario: AGRGBO

Libreria.....: _____ Fecha....: _____
 Archivo fuente.....: _____ Hora.....: _____
 Programa.....: _____

Sl	Libreria	A.Fuente	Programa	Fecha	Hora	Perfo	Calidad	B.prac	M.Prac	Segur.	Complej.	Obsol.
—	LIBGBO	FUENTES1	AG0011	2017-05-08	14.57.08	54	33	47	27	100	50	60
—	LIBGBO	FUENTES1	AG0011A	2017-05-08	14.57.08	54	34	50	27	100	50	60
—	LIBGBO	FUENTES1	AG0012	2017-05-08	14.57.08	47	30	47		100	53	65
—	LIBGBO	FUENTES1	AG0121	2017-05-08	14.57.08	50	39	57		100	53	65
—	LIBGBO	FUENTES1	AG0122	2017-05-08	14.57.08	50	31	46	14	100	66	65
—	LIBGBO	FUENTES1	AI0008	2017-05-08	14.57.08	53	46	50	75	100	58	68
—	LIBGBO	FUENTES1	AI0077	2017-05-08	14.57.08	61	40	61	16	100	63	65
—	LIBGBO	FUENTES1	BAP003	2017-05-08	14.57.08	100	100	100	50	100	100	
—	LIBGBO	FUENTES1	BA0010	2017-05-08	14.57.08	42	23	35	33	100	46	57
—	LIBGBO	FUENTES1	BA0020	2017-05-08	14.57.08	45	28	38	33	100	46	83
—	LIBGBO	FUENTES1	BA0030	2017-05-08	14.57.08	47	27	40	33	100	46	50
—	LIBGBO	FUENTES1	BA0031	2017-05-08	14.57.08	50	27	42	33	100	46	50
—	LIBGBO	FUENTES1	BA0040	2017-05-08	14.57.08	52	31	47	37	100	50	50
—	LIBGBO	FUENTES1	BA0050	2017-05-08	14.57.08	44	29	37	42	100	46	54
—	LIBGBO	FUENTES1	BA0055	2017-05-08	14.57.08	63	38	58	28	100	64	57

F3=Salir F10=Ayuda

Figura 101. Consulta general de calificación.

William Flores
1037-18



ESCUELA DE POSGRADO

UNIVERSIDAD CÉSAR VALLEJO

FORMATO DE SOLICITUD

SOLICITA:

Gestión de I.A.
visto bueno de I.A.
tesis

ESCUELA DE POSGRADO

Giovanni Barreto Ortiz con DNI N° 48839223
(Nombres y apellidos del solicitante) (Número de DNI)

domiciliado (a) en cll. los electricistas 135 I.A. Tolva
(Calle / Calle / No. / Urb. / Distrito / Provincia / Región)

ante Ud. con el debido respeto expongo lo siguiente:

Que en mi condición de alumno de la promoción: del programa: Gestión de
tecnología de inf. identificado con el código de matrícula N° 7000955576
(Promoción) (Nombre del programa) (Código de alumno)

de la Escuela de Posgrado, recorro a su honorable despacho para solicitarle lo siguiente:

Se realice la gestión del visto bueno de mi tesis "Sistema Qsource en la calidad del software desarrollada en RPB"

Por lo expuesto, agradeceré que me atiendan mi petición por ser de justicia.



Lima, 23 de Junio de 2018

Giovanni Barreto Ortiz
Hora: / Firma: (Firma del solicitante)

Documentos que adjunto:

- a. Tesis original
- b. C.R.P.A. y C.S. d.i.r. suscri.
- c. C.R.P.A. D.I.C.I.M.E.N. SUSCR.
- d. C.R.P.A. ACTA Aprob. vig.

Cualquier consulta por favor comunicarse conmigo al:

Teléfonos: 974604104
Email: g.barreto@ucv.edu.pe



Dr. William Sebastián Flores Sobado
Docente Investigador de Posgrado
CEEL N° 09/26

VB para entrega
[Handwritten signature]



Acta de Aprobación de originalidad de Tesis

Yo, Willian Sebastian Flores Sotelo, docente de la Escuela de Posgrado de la Universidad César Vallejo filial Lima Norte, revisor de la tesis titulada "**SISTEMA QSOURCE EN LA CALIDAD DEL SOFTWARE DESARROLLADO EN RPG**" del (de la) estudiante **Giovanni Barrero Ortiz**, constato que la investigación tiene un índice de similitud de 20% verificable en el reporte de originalidad del programa Turnitin.

El/la suscrito(a) analizo dicho reporte y concluyo que cada una de las coincidencias detectadas no constituye plagio. A mi leal saber y entender la tesis cumple con todas las normas para el uso de citas y referencias establecidas por la Universidad César Vallejo.

Lima, 18 de marzo del 2018

Dr. Willian Sebastian Flores Sotelo
Asesor de la tesis



UNIVERSIDAD CÉSAR VALLEJO

Centro de Recursos para el Aprendizaje y la Investigación (CRAI)
"César Acuña Peralta"

FORMULARIO DE AUTORIZACIÓN PARA LA PUBLICACIÓN ELECTRÓNICA DE LAS TESIS

1. DATOS PERSONALES

Apellidos y Nombres: (solo los datos del que autoriza)

.....BARRERO ORTIZ.....GIOVANNI.....

D.N.I. :4883723.....

Domicilio :CALLE LAS ELECTRICISTAS N.5 LA TRINIDAD

Teléfono : Fijo : 6239705 Móvil : 974604104

E-mail :g.barrera@hotmail.com.....

2. IDENTIFICACIÓN DE LA TESIS

Modalidad:

Tesis de Pregrado

Facultad :

Escuela :

Carrera :

Título :

Tesis de Post Grado

Maestría

Doctorado

Grado :MAESTRÍA.....

Mención :GESTIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN.....

3. DATOS DE LA TESIS

Autor (es) Apellidos y Nombres:

.....BARRERO ORTIZ GIOVANNI.....

.....

.....

Título de la tesis:

.....Sistema Qsource en la Calidad del

.....SOFTWARE DESARROLLADO EN RPG.....

Año de publicación :2018.....

4. AUTORIZACIÓN DE PUBLICACIÓN DE LA TESIS EN VERSIÓN ELECTRÓNICA:

A través del presente documento, autorizo a la Biblioteca UCV-Lima Norte,
a publicar en texto completo mi tesis.

Firma :

Giovanni Barrero Ortiz


Fecha :

25/07/2018

Feedback Studio - Google Chrome

feedback studio

Tesis: BARRERO ORTIZ GIOVANNI 20180218



ESCUELA DE POSGRADO
UNIVERSIDAD CÉSAR VALLEJO

Sistema QSOURCE en la Calidad del software
Desarrollado en RPG

TESIS PARA OPTAR EL GRADO ACADÉMICO DE:
Maestro en Gestión de Tecnologías de Información

AUTOR:
Br. Giovanni Barrero Ortiz

ASESOR:
Dr. Willian Sebastián Flores Sotelo

SECCIÓN:
Ingeniería

LÍNEA DE INVESTIGACIÓN:
Sistemas Basados en Gestión de Procesos de Negocio

PERÚ - 2018

Resumen de coincidencias

20 %

1	Entregado a Universidad	1 %
2	Cyberfesta.universia.edu	1 %
3	Entregado a Universidad	1 %
4	www.univ.edu	1 %
5	Publicaciones	1 %
6	Entregado a Universidad	1 %
7	www.univ.com	1 %
8	www.univ.edu	<1 %
9	www.univ.edu	<1 %
10	repositorio.universia.edu	<1 %

Página 1 de 114 Número de palabras: 400/500