**Plantilla para la sección de desarrollo**

**Plan de Calidad de Proyecto**

**PLAN DE GESTIÓN DE LA CALIDAD DEL PROYECTO**

1. Roles y responsabilidades (respecto al Plan de Gestión Calidad)

*Esta sección sienta los roles y responsabilidades en relación con la gestión de la calidad del proyecto de forma específica.*

|  |  |
| --- | --- |
| **Rol** | **Responsabilidades** |
| Planeación | Ayudar al equipo a ejecutar un proyecto bien planificado y seguido.  Garantizar que el equipo trabaje con un plan definido y documentado. |
| Proceso | Garantizar que el equipo tenga procesos definidos y documentados para las actividades de desarrollo, administración y funciones del equipo. |
| Calidad | Liderar al equipo en la producción y seguimiento de los parámetros de calidad para el proyecto.  Proporcionar un análisis oportuno y advertir sobre problemas de calidad. |
| Soporte | Garantizar que el equipo tenga las herramientas y métodos necesarios para hacer el proyecto.  Manejar las funciones de gestión de configuración y control de cambios del equipo. |
| Interfaz con el cliente | Mantener un enfoque en las necesidades del cliente.  Liderar al equipo en la producción, refinación y verificación de los requisitos del producto.  Establecer estándares y procedimientos en el equipo. |
| Diseño | Liderar al equipo en la producción, refinación y verificación del diseño.  Identificar y resolver todos los problemas de diseño, documentar y confirmar las resoluciones. |
| Implementación | Asegurar que la implementación se ajuste completamente al diseño.  Establecer los estándares y procedimientos que el equipo usará para la implementación del producto.  Asegurarse que el equipo tenga estándares de codificación, conteo de LOC, lenguaje de programación y documentación. |
| Pruebas | Definir y planificar las pruebas del sistema durante la fase de diseño.  Definir y planificar las pruebas de integración durante la fase de implementación.  Analizar datos de cada fase de prueba para identificar elementos del producto que sean propensos a defectos.  Identificar las áreas que necesitan inspección o pruebas. |

1. Política de Calidad del Proyecto (enfoque para gestión)

*En esta sección se establecen los lineamientos y directrices generales para la gestión de la calidad del proyecto. Se realiza desde tres perspectivas: enfoque para la (1) planificación de la calidad del proyecto, (2) enfoque para el aseguramiento de la calidad del proyecto, (3) enfoque para el control de la calidad del proyecto.*

|  |
| --- |
| Los productos línea base tienen que pasar por un proceso de inspección.  Utilizar herramientas para calcular las métricas.  Aplicar pruebas unitarias, integración y de sistema. |

1. Línea Base de Calidad (factores y métricas)

*En esta sección se establecen las métricas detallas y la forma en que será medido el proyecto, producto, servicio o resultado del proyecto. Las métricas son insumos para el aseguramiento de la calidad (donde se valida que los procesos podrán lograrlas) y para el control de la calidad (donde se comparan resultados contra métricas para verificar cumplimiento y definir si es necesaria la toma de acciones correctivas). La información de stakeholders y requisitos del proyecto (así como su priorización) son insumos para la definición de las métricas. Los riesgos son también un insumo a considerar.*

1. Factores de éxito para la calidad (de acuerdo con la priorización de requisitos del proyecto).

|  |
| --- |
| Pruebas planificadas, inspecciones, seguimiento de estándares. |

1. Línea Base de Calidad (métricas)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Objetivo de Calidad | Métrica (s) | Definición de la métrica (método de medición) | Resultado esperado | Frecuencia de medición | Responsable del cumplimiento de la métrica |
| Mantener un número de defectos estandarizado por cada KLOC. | Densidad de defectos (defect/KLOC) | A través de los registros de defectos y tamaño. | 14 defectos por KLOC | Cada dos semanas | GBJA |
| Descubrir al menos el 60% de defectos en las revisiones. | Yield | Defectos removidos en las revisiones. Se mide en base a los registros de supresión de defectos. | %65 o más | Cada dos semanas | GBJA |
| Agilizar y optimizar los procesos de revisión. | Velocidad de revisión | Rapidez con la que se revisan ciertas LOC/Hr. | Código: 200 LOC por hora.  Documentos: 4 páginas por hora | Cada dos semanas | GBJA |
| Disminuir el número de defectos para aumentar la calidad del producto. | A/FR | Mide el costo de evaluación y el costo de fallas. Un alto A/FR está asociado con un bajo número de defectos y una alta calidad del producto. | 2.0 o más | Cada dos semanas | GBJA |
| Encontrar en qué etapa se encuentran y se eliminan más defectos tomando en cuenta nuestros datos anteriores. | DRL | Mide la efectividad relativa de supresión de defectos en las fases de revisión de diseño, revisión de código y compilación | 1.28 en promedio. | Cada dos semanas | GBJA |

1. Plan de aseguramiento y control (actividades de calidad)

*En esta sección se establecen las actividades orientadas a asegurar que se cumplan los objetivos y métricas de calidad, y por tanto, los requisitos del proyecto. Deben establecerse tanto acciones de aseguramiento (costos de calidad preventivos) como acciones de control (costos de calidad de detección) para que la gestión de calidad se desarrolle de forma integral. Además de las actividades, deberá incluir un mínimo de tres documentos para la calidad (plantillas, diagramas, hojas de control, instrucciones, formularios o similares) que serán necesarios para generar los registros de calidad del proyecto.*

1. Factores de éxito para la calidad (de acuerdo con la priorización de requisitos del proyecto)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entregable** | **Requisito** | **Actividades de aseguramiento y control** | **Frecuencia** | **Responsable** |
| Diseños | Conocer los requerimientos del proyecto. | Aseguramiento: Tener reuniones con el cliente.  Hacer diagramas de procesos y eventos. | Cada día, hasta que se terminen las etapas de diseño. | ASCG |
| Control:  Hacer revisiones de diseño. | Cada día, hasta que se terminen las etapas de diseño. | Equipo. |
| Código | Cumplir con el estándar de codificación. | Aseguramiento: Generar un script de estándar de codificación. | Cada día, hasta que terminen las etapas de codificación | GDE |
| Control: aplicar pruebas unitarias y de sistema.  Hacer revisiones de código. | Cada día, hasta que terminen las etapas de codificación | Equipo |
| Planes | Basarse en datos históricos. | Aseguramiento:  Utilizar datos de planes individuales para programar planes y tareas.  Definir un plan y documentarlo. | Una vez, al realizar el lanzamiento. | CSFO |
| Control:  Revisar el plan y si es necesario hacer cambios. | Cada vez que cambie el cronograma del proyecto o cada que se presente algún riesgo | CSFO |
| Pruebas | Asegurarse que el código haya sido compilado y ejecutado exitosamente. | Aseguramiento:  Analizar datos de cada fase de prueba para identificar elementos propensos a defectos. | Semanalmente | SMME |
| Control:  Realizar pruebas unitarias, integración y de sistema. | Cada día, hasta que terminen las fases de pruebas. | Equipo |
|  |  | Aseguramiento: |  |  |
| Control: |  |  |

1. Documentos para la calidad (incluya aquí al menos 3 ejemplos)

Ejemplo 1.- Estándar de codificación:

|  |  |
| --- | --- |
| Comentarios. | Si se va a describir un método el comentario debe de ir antes de la declaración del método. Ejemplo:  //------------- Método [1] -------- public int DesvStd() { }  Cuando solo se vaya a comentar una línea, el comentario se debe de poner arriba de la línea de código que deberá describir y dejando un espacio entre el comentario y las diagonales (//). Ejemplo:  // sacando raíz double Y = math.sqrt(X,3);  Ejemplo de cómo no se usará:  double Y = math.sqrt(X,3); //sacando raíz |
| Uso de llaves. | Cada llave ocupará una línea de código (no contable) sin necesidad de juntarlas con la línea de declaración del método. Ejemplo:  public double ObtMayor()  {  }  public int SacarMedia()  {  }  Ejemplos de cómo no se usará:  public double ObtMayor() {  }  public int SacarMedia() {  } |
| Declaración de variables. | Para declarar una variable se debe hacer línea por línea, puede ir con comentario o no y el nombre de ellas debe de empezar con una mayúscula, si tiene un nombre compuesto de dos o más palabras, cada palabra debe de iniciar con mayúscula y abreviado si la palabra es larga. Todas las variables deberán ser inicializadas en un constructor. Ejemplos:  int Sumar; double DesvEst; String NomEstud;  Ejemplos de cómo no se usará:  int Sumar = 0; int y = 1;  String Nombre, ApellidoPaterno, ApellidoMaterno. |
| Declaración de clases. | Las clases en el momento de que sean nombradas deberán empezar con una mayúscula, si se conjuntan dos palabras cada una iniciará con mayúscula y será abreviada si la palabra es larga. Cada clase llevará un comentario en la parte superior “//------ Clase [n] -------” Ejemplo:  //------ Clase [1] -------  public class Operaciones  {  }  Ejemplos de cómo no se usará:  public class Operaciones {  } |
| Declaración de métodos. | Los métodos su declaración es similar que las variables empiezan con una mayúscula y de igual manera si se conjuntan dos palabras cada una iniciará con mayúscula y será abreviada si la palabra es larga. Cada método llevará un comentario en la parte superior “//------ Metodo [n] -------” Ejemplo:  //------ Metodo [1] ------- public double ObtMayor() { }  //------ Metodo [2] ------- public int SacarMedia() { }  //------ Metodo [3] ------- public ArrayList ObtListaEst(int NumCtrl) { }  Ejemplos de cómo no se usará:  public double ObtenerMayor() { }  public ArrayList ObtenerListaEstudiante(int NumeroControl) { } |
| Nomenclatura de fórmulas. | Cuando se va a realizar una expresión matemática, se deben de usar los espacios para ver con claridad los operadores y operandos. Ejemplo:  Suma = X + P;  AreaTrian = (B \* H) / 2;  Ejemplos de cómo no se usará:  Suma=X+P;  AreaTrian=(B\*H)/2; |
| Indentación. | La indentación se aplica con una tabulación después de la posición de la llave del método o clase, así para todas las líneas de código que pertenezcan a dicho método o clase. Ejemplo;  for (i = 0; i < 10; i ++)  {  if (i == 7)  {  printf(“Es un siete”);  }  }  Ejemplos de cómo no se usará:  for (i = 0; i < 10; i ++) {  if (i == 7)  {  printf(“Es un siete”);  }  } |
| Concatenación. | Para imprimir un mensaje de texto junto con una variable, se utilizarán espacios entre la cadena a imprimir, el operando ‘+’ y la variable. Ejemplo:  JOptionPane.showMessageDialog(null, “El numero es: ” + Numero);  Ejemplos de cómo no se usará:  JOptionPane.showMessageDialog(null,“El numero es: ”+Numero); |
| Declaración de objetos. | Al momento de declarar un objeto se utilizará la letra ‘m’ seguida del nombre de la clase para identificarlo. Ejemplo:  Persona mPersona = new Persona();  Auto mAutoRojo = new Auto(); |
| Espacios en blanco. | Entre cada método habrá un espacio en blanco con la finalidad de mostrarlo de manera clara. Ejemplo:  public int Sumar(int X, int Y)  {  return x + y;  }  public int Restar(int X, int Y)  {  return x - y;  }  Ejemplos de cómo no se usará:  public int Sumar(int X, int Y)  {  return x + y;  }  public int Restar(int X, int Y)  {  return x - y;  } |
| Líneas modificadas. | Para conocer si se modificó una línea, después del punto y coma se deberá dejar un espacio de tabulación y colocar un pequeño comentario “//M” indicando que esa línea fue modificada. Ejemplo:  for (i = 0; i < 10; i ++)  {  if ( (i == 3) || (i== 7) || (i==9))  {  printf(“Es tres, siete ó es un nueve”); //M  }  } |
| Líneas añadidas. | Para conocer si se añadió una línea a una clase ya existente, después del punto y coma se deberá dejar un espacio de tabulación y colocar un pequeño comentario “//A” indicando que esa línea fue añadida. Ejemplo:  for (i = 0; i < 10; i ++)  {  if ( (i == 3) || (i== 7) || (i==9))  {  printf(“Es tres, siete ó es un nueve”); //M  }  else //A  {  printf(“NO es tres, siete ó es un nueve”); //A  }  } |

Ejemplo 2.- Estándar de conteo de LOC:

|  |  |  |
| --- | --- | --- |
| **Tipo de conteo.** | **Tipo.** | **Comentario.** |
| Physical/Logical | Lógicas. |  |
| **Tipo de declaración.** | **Incluído.** | **Comentario.** |
| Ejecutable | Si. |  |
| No Ejecutable: | No. |  |
| Declaraciones | Si. | Incluidas en las notas 1 y 2. |
| Directivas de compilación. | Si. | Incluidas en la nota 3. |
| Comentarios | No. |  |
| Líneas en blanco.  Conversión de líneas físicas a lógicas. | No. |  |
| Si | Por cada símbolo de lógico OR (||) y AND (&&) se contará una vez más como si fuera una línea de código. Ejemplo:  for (i = 0; i < 10; i ++)  {  if ( (i == 3) || (i== 7) || (i==9))  {  printf(“Es tres, siete ó es un nueve”);  }  }  Total de líneas: 5.  Ejemplos de cómo no se usará:  for (i = 0; i < 10; i ++)  {  if ( (i == 3) || (i== 7) || (i==9))  {  printf(“Es tres, siete ó es un nueve”);  }  }  Total de líneas: 3. |
| Otros elementos. | No. |  |
|  |  |  |
| **Aclaraciones.** |  | **Ejemplos/Casos.** |
| Declaraciones vacías. | Si. | Incluida en nota 1. |
| Sentencias de  expresiones. | Si. | Contar una vez. |
| Símbolos finales. | No. |  |
| Palabras Clave. | Si. | Contar una vez cada que aparezcan. |
| Etiquetas. | No. |  |
|  |  |  |
| **Notas.** |  |  |
| Nota 1. |  | Contar declaración de parámetro cuando se encuentre. |
| Nota 2. |  | Contar solo una vez cada que aparezcan las siguientes palabras:  CASE, DO, ELSE, ENUM, FOR, IF, PRIVATE, PUBLIC, SWITCH, WHILE, NULL, THIS, BREAK, RETURN, TRY, CATCH, EXCEPTION. |
| Nota 3. |  | Contar una vez cada que aparezcan las palabras ‘package’, ‘import’. |

Ejemplo 3.- Plantilla de diseño funcional:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Student** | | |  | | **Date** |  |
| **Program** | | |  | | **Program #** |  |
| **Instructor** | | |  | | **Language** |  |
|  | | | | | | |
| **Class Name** | |  | | | | |
| **Parent Class** | |  | | | | |
|  | |  | | | | |
|  | |  | | | | |
|  | |  | | | | |
|  | | | | | | |
| **Attributes** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  |  | | |  | | |
|  |  | | |  | | |
|  |  | | |  | | |
|  |  | | |  | | |
|  |  | | |  | | |
|  |  | | |  | | |
|  | | | | | | |
| **Items** | | | | | | |
|  | **Declaration** | | | **Description** | | |
|  |  | | |  | | |
|  |  | | |  | | |
|  |  | | |  | | |
|  |  | | |  | | |
|  |  | | |  | | |
|  |  | | |  | | |
|  |  | | |  | | |