

**Título:** Desarrollo de una herramienta de diagnóstico CMMI Nivel 2

**Asignatura:** Gestión de Proyectos Software

**Profesora:** Celia Gutiérrez Cosío

**Fecha de entrega:** 22 de octubre de 2025

**Integrantes del equipo:**

* Nasratullah Jabarkhil Jabarkhil
* Diego Santos Muñoz
* Anderson

Table of Contents

[**1. INTRODUCCIÓN** 2](#_Toc211515319)

[**Objetivo de la práctica** 2](#_Toc211515320)

[**KPAs evaluadas (Nivel 2 de CMMI)** 2](#_Toc211515321)

[**Requisitos implementados** 2](#_Toc211515322)

[**2. VERSIÓN 1: CÓDIGO DESARROLLADO SIN IA** 3](#_Toc211515323)

[**Descripción** 3](#_Toc211515324)

[**Enfoque inicial** 3](#_Toc211515325)

[**Estructura de archivos (Versión 1)** 3](#_Toc211515326)

[**Características principales** 3](#_Toc211515327)

[**Dificultades encontradas** 4](#_Toc211515328)

[**Tiempo de desarrollo** 4](#_Toc211515329)

[**3. VERSIÓN 2: CÓDIGO DESARROLLADO CON IA** 5](#_Toc211515330)

[**3.1. Herramienta de IA utilizada** 5](#_Toc211515331)

[**3.2. Prompts y respuestas realizadas con la herramienta de IA** 5](#_Toc211515332)

[**4. DIFERENCIAS ENTRE VERSIÓN 1 Y VERSIÓN 2** 6](#_Toc211515333)

[**Tabla comparativa** 6](#_Toc211515334)

[**Diferencias funcionales detalladas** 6](#_Toc211515335)

[**5. COMPARACIÓN Y CONCLUSIONES** 8](#_Toc211515336)

[**¿Cuál versión es mejor y por qué?** 8](#_Toc211515337)

[**Ventajas del uso de IA en el desarrollo** 8](#_Toc211515338)

[**Desventajas o limitaciones observadas** 9](#_Toc211515339)

[**Conclusión final** 9](#_Toc211515340)

**1. INTRODUCCIÓN**

Este documento presenta la memoria de la Práctica 1 de la asignatura de Gestión de Proyectos Software, que consiste en el desarrollo de una herramienta de diagnóstico para evaluar el cumplimiento de las KPAs (Key Process Areas) del Nivel 2 del modelo CMMI en proyectos de desarrollo de software.

**Objetivo de la práctica**

Diseñar y desarrollar una herramienta software que permita:

* Diagnosticar el grado de cumplimiento de las KPAs del nivel 2 del modelo CMMI
* Proporcionar recomendaciones automáticas para mejorar los procesos deficientes

**KPAs evaluadas (Nivel 2 de CMMI)**

1. Gestión de requisitos
2. Planificación de proyectos
3. Seguimiento y control de proyectos
4. Gestión de configuración
5. Aseguramiento de calidad

**Requisitos implementados**

La herramienta desarrollada cumple con los 9 requisitos funcionales establecidos:

1. Formulario de evaluación por KPA (preguntas Sí/No/Parcial)
2. Cálculo del nivel de cumplimiento por KPA (diagnóstico automático)
3. Generación de recomendaciones automáticas por KPA
4. Informe final por KPA
5. Evaluación para todas las KPA (resumen de resultados)
6. Diagnóstico general para todas las KPA del Nivel 2
7. Verificación de cumplimiento del Nivel 2
8. Recomendaciones para alcanzar el nivel 2
9. Conclusión con fecha y hora

**2. VERSIÓN 1: CÓDIGO DESARROLLADO SIN IA**

**Descripción**

La Versión 1 fue desarrollada completamente por los miembros del equipo sin ayuda de herramientas de Inteligencia Artificial.

**Enfoque inicial**

Para la Versión 1, comenzamos diseñando la arquitectura del sistema de forma manual. Decidimos crear una interfaz de línea de comandos (CLI) en Python, dividiendo el código en módulos separados para mantener una estructura clara y modular.

El proceso de desarrollo fue:

1. Análisis de requisitos del enunciado
2. Diseño de la estructura de datos (diccionarios para KPAs y respuestas)
3. Implementación de funciones básicas de evaluación
4. Desarrollo del sistema de cálculo de porcentajes
5. Implementación del sistema de recomendaciones
6. Pruebas manuales

**Estructura de archivos (Versión 1)**

* diagnostico\_cmmi\_nivel2.py - Programa principal CLI
* KPAS.py - Diccionario con preguntas por KPA
* VALOR\_RESPUESTA.py - Mapeo de respuestas a valores
* RECOMENDACIONES\_BASE.py - Recomendaciones predefinidas
* porcentaje.py - Función de clasificación

**Características principales**

* Interfaz de línea de comandos interactiva
* Sistema de menú con 3 opciones:
  1. Evaluar todas las KPAs
  2. Evaluar una KPA específica
  3. Salir
* Cálculo automático de cumplimiento (porcentaje)
* Clasificación según criterios CMMI:
  + ≥ 80% → Implementada
  + 50-79% → Parcialmente implementada
  + < 50% → Deficiente
* Generación de recomendaciones contextuales

**Dificultades encontradas**

Las principales dificultades fueron:

* Diseñar el flujo de navegación del menú
* Implementar la lógica de recomendaciones específicas por pregunta
* Evitar duplicación de código
* Manejo de entrada de usuario y validación

**Tiempo de desarrollo**

Aproximadamente 6-8 horas de trabajo, distribuidas en:

* 2 horas: análisis y diseño
* 4 horas: implementación
* 1-2 horas: pruebas y correcciones

**3. VERSIÓN 2: CÓDIGO DESARROLLADO CON IA**

**3.1. Herramienta de IA utilizada**

**Herramienta:** ChatGPT

**Versión:** ChatGPT-5

**3.2. Prompts y respuestas realizadas con la herramienta de IA**

A continuación, se presentan los principales prompts utilizados y las respuestas obtenidas:

***PROMPT 1: Solicitud inicial de estructura***

**Prompt enviado y respuesta recibida:**

A screenshot of a chat

AI-generated content may be incorrect.

**Conversación completa con IA**

<https://chatgpt.com/share/68f0d9b9-a720-8003-aa3e-fe34dcb28b7c>

**Tiempo de desarrollo con IA**

Aproximadamente 3-4 horas de trabajo, distribuidas en:

* 5 minutos: Darle el prompt y el código
* 5 minutos: implementación del código por la IA

**4. DIFERENCIAS ENTRE VERSIÓN 1 Y VERSIÓN 2**

**Tabla comparativa**

| Aspecto | Versión 1 (Sin IA) | Versión 2 (Con IA) |
| --- | --- | --- |
| **Interfaz** | Solo CLI (consola) | CLI + GUI (Tkinter) |
| **Líneas de código** | ~250 líneas | ~610 líneas (ambas versiones) |
| **Modularización** | Básica | Alta (mejor separación) |
| **Documentación** | Comentarios básicos | Docstrings + comentarios detallados |
| **Manejo de errores** | Básico | Mejorado con validaciones |
| **Experiencia de usuario** | Funcional, texto plano | Intuitiva, gráfica y textual |
| **Recomendaciones** | Generales por KPA | Generales + específicas por pregunta |
| **Tiempo desarrollo** | 6-8 horas | 10 minutos |
| **Curva de aprendizaje** | Lenta, investigación manual | Rápida, guiada por IA |

**Diferencias funcionales detalladas**

*1. Interfaz de usuario*

**Versión 1:**

* Solo interfaz de texto en terminal
* Menú simple con números
* Salida en consola

**Versión 2:**

* Interfaz gráfica con ventanas
* Botones y formularios visuales
* Scroll automático para contenido largo
* Mejor experiencia visual

*2. Estructura del código*

**Versión 1:**

* Un archivo principal (diagnostico\_cmmi\_nivel2.py)
* Funciones básicas de evaluación
* Código funcional pero menos estructurado

**Versión 2:**

* Dos archivos principales (CLI + GUI)
* Clase CMMIApp para la GUI
* Mejor separación de responsabilidades
* Reutilización de funciones

*3. Sistema de recomendaciones*

**Versión 1:**

* Recomendaciones base por KPA
* Mensaje genérico si todas las respuestas son “Sí”

**Versión 2:**

* Recomendaciones base + específicas por pregunta fallida
* Mensajes diferenciados para “No” vs “Parcial”
* Sistema anti-duplicados mejorado

*4. Flujo de evaluación*

**Versión 1:**

* Evaluación lineal
* Sin indicador de progreso

**Versión 2:**

* Opción de cancelar en cualquier momento
* Mejor gestión del estado de la aplicación

**5. COMPARACIÓN Y CONCLUSIONES**

**¿Cuál versión es mejor y por qué?**

**Versión elegida como mejor: Versión 2 (Con IA)**

*Justificación:*

1. **Mayor funcionalidad**
   * La Versión 2 ofrece dos interfaces (CLI y GUI), dando más opciones al usuario
   * La GUI hace la herramienta más accesible para usuarios no técnicos
2. **Mejor experiencia de usuario**
   * La interfaz gráfica es más intuitiva
   * El sistema de scroll permite manejar muchas preguntas cómodamente
3. **Código más mantenible**
   * Mejor documentación con docstrings
   * Estructura más modular facilita futuras mejoras
   * Separación clara entre lógica y presentación
4. **Mayor valor educativo**
   * Aprendimos sobre Tkinter y GUI en Python
   * Entendimos mejor las arquitecturas modulares
   * Practicamos el uso efectivo de IA como herramienta
5. **Eficiencia en el desarrollo**
   * Menor tiempo de desarrollo (casi 60% menos)
   * Menos errores gracias a las sugerencias de la IA
   * Código más robusto desde el inicio

**Ventajas del uso de IA en el desarrollo**

1. **Aceleración del desarrollo**
   * Reducción del 40-50% en tiempo de implementación
   * Soluciones más rápidas a problemas técnicos
2. **Mejora de la calidad del código**
   * Sugerencias de mejores prácticas
   * Código más idiomático (pythonic)
   * Menos bugs iniciales
3. **Aprendizaje acelerado**
   * Exposición a patrones de diseño avanzados
   * Explicaciones de conceptos nuevos (ej: Tkinter)
   * Ejemplos prácticos inmediatos
4. **Capacidad de implementar más funcionalidades**
   * Sin IA, solo habríamos hecho la CLI
   * Con IA, pudimos añadir la GUI completa

**Desventajas o limitaciones observadas**

1. **Dependencia de la herramienta**
   * Riesgo de no entender completamente el código generado
   * Necesidad de validar y probar las sugerencias
2. **Necesidad de supervisión**
   * No todo el código sugerido funciona a la primera
   * Requiere conocimiento para adaptar las respuestas
3. **Curva de aprendizaje en el uso de IA**
   * Aprender a hacer buenos prompts es importante
   * Necesidad de iterar varias veces

**Conclusión final**

El uso de Inteligencia Artificial como herramienta de apoyo en el desarrollo de software ha demostrado ser muy beneficioso para este proyecto. La Versión 2, desarrollada con ayuda de IA, es superior en funcionalidad, usabilidad y calidad de código.

Sin embargo, es fundamental destacar que la IA es una **herramienta de apoyo**, no un reemplazo del desarrollador. Se requiere:

* Conocimiento base de programación
* Capacidad de análisis crítico
* Habilidad para validar y adaptar el código
* Comprensión de los requisitos del proyecto

La combinación de habilidades humanas + asistencia de IA resulta en el mejor resultado posible: desarrollo más rápido, código de mayor calidad y aprendizaje acelerado.