Cristian Suárez
Lógica de Programación
Universidad Internacional

Piedra, Papel o Tijera

Diagrama de Componentes

Pasos para desarrollar el Juego Piedra, papel o tijera.

Paso 1: Definición del Problema (El "Qué") Debemos definir claramente qué queremos que haga el sistema para evitar ambigüedades.

Objetivo: Crear un sistema que permita un juego de azar simple entre un usuario humano y la computadora (IA). Condición de Victoria: El sistema debe decidir un ganador basándose en reglas preestablecidas de jerarquía circular.

Alcance (Scope): Es un juego de 1 contra 1 (Usuario vs. CPU). Es una sola ronda por ejecución (o un bucle hasta que el usuario decida salir). No habrá apuestas ni multijugador en red por ahora.


1
PASO 1: DEFINICIÓN DEL PROBLEMA
(El "Qué")
Ambigüedad
OBJETIVO:
Juego simple de azar,
Usuario vs. CPU (IA)
[cite: 1]
REGLAS
CIRCULARES
Condición
de Victoria
[cite: 1]
[cite: 1]
[cite: 1]
USUARIO
CPU (IA) [cite: 1]
ALCANCE (SCOPE) [cite: 1]
1v1
Competa sola
jugador 1v1
sola ronda/
bucle [cite: 1]
No apuestas
Multijugador

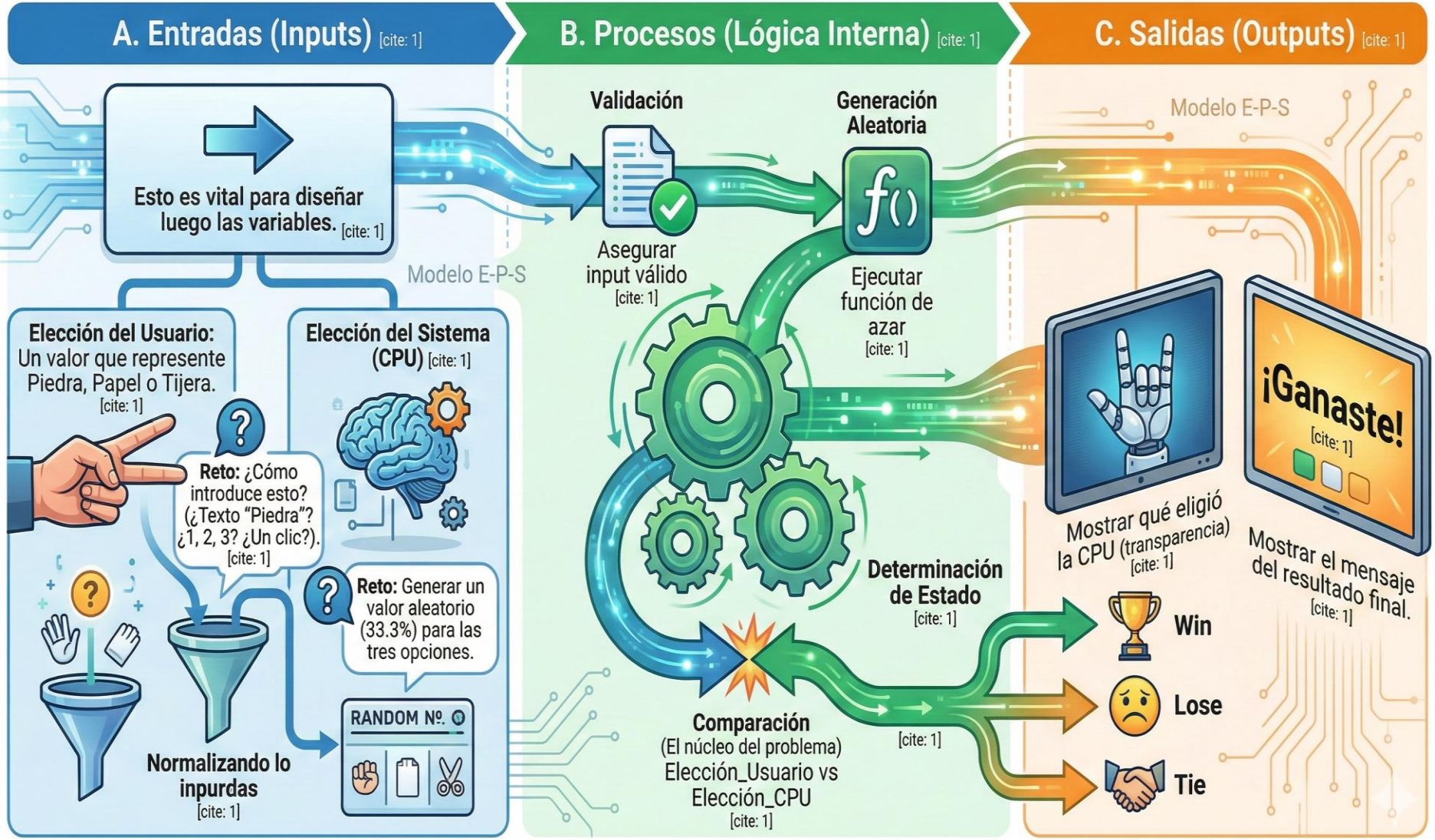
Paso 2: Análisis de Datos (Modelo E-P-S) Aquí identificamos qué información entra, cómo se transforma y qué sale. Esto es vital para diseñar luego las variables.

A. Entradas (Inputs) - ¿Qué necesita el sistema? Elección del Usuario: Un valor que represente Piedra, Papel o Tijera. Reto a resolver: ¿Cómo introduce esto el usuario? (¿Texto "Piedra"? ¿Un número 1, 2, 3? ¿Un clic?). Debemos normalizar esta entrada. Elección del Sistema (CPU): Reto a resolver: La computadora no "piensa". Necesitamos generar un valor aleatorio que represente una de las tres opciones con igual probabilidad (33.3%).

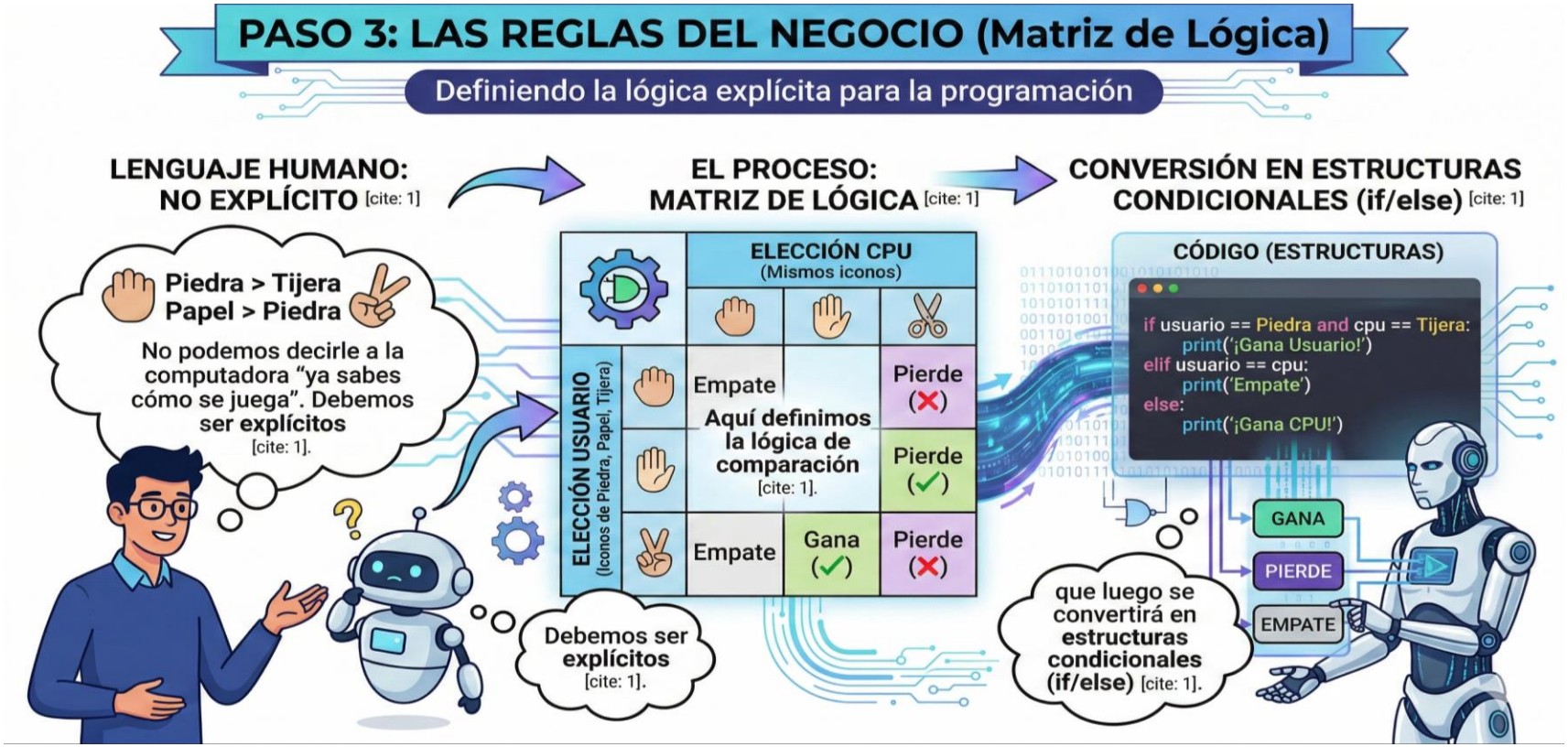
B. Procesos (Lógica Interna) - ¿Qué hace el sistema? Validación: Asegurar que lo que ingresó el usuario sea válido (ejem: que no escriba "Fuego"). Generación Aleatoria: Ejecutar la función de azar para la CPU. Comparación (El núcleo del problema): Confrontar la Elección Usuario vs Elección CPU. Determinación de Estado: Asignar el resultado: Ganar, Perder o Empatar.

C. Salidas (Outputs) - ¿Qué ve el usuario? Mostrar qué eligió la CPU (para dar transparencia al juego). Mostrar el mensaje del resultado final ("¡Ganaste!", "Empate", etc.).

Paso 2: Análisis de Datos (Modelo E-P-S) [cite: 1]



Paso 3: Las Reglas del Negocio (Matriz de Lógica) En programación, no podemos decirle a la computadora "ya sabes cómo se juega".
Debemos ser explícitos. Aquí definimos la lógica de comparación que luego se convertirá en estructuras condicionales (if/else).



Paso 4: Identificación de Restricciones y Casos Borde Un buen análisis de problemas anticipa dónde puede fallar el software.

Entradas Inválidas (Dirty Data):

Problema: El usuario escribe "Roca" en lugar de "Piedra" o escribe "Piedra " (con espacio).

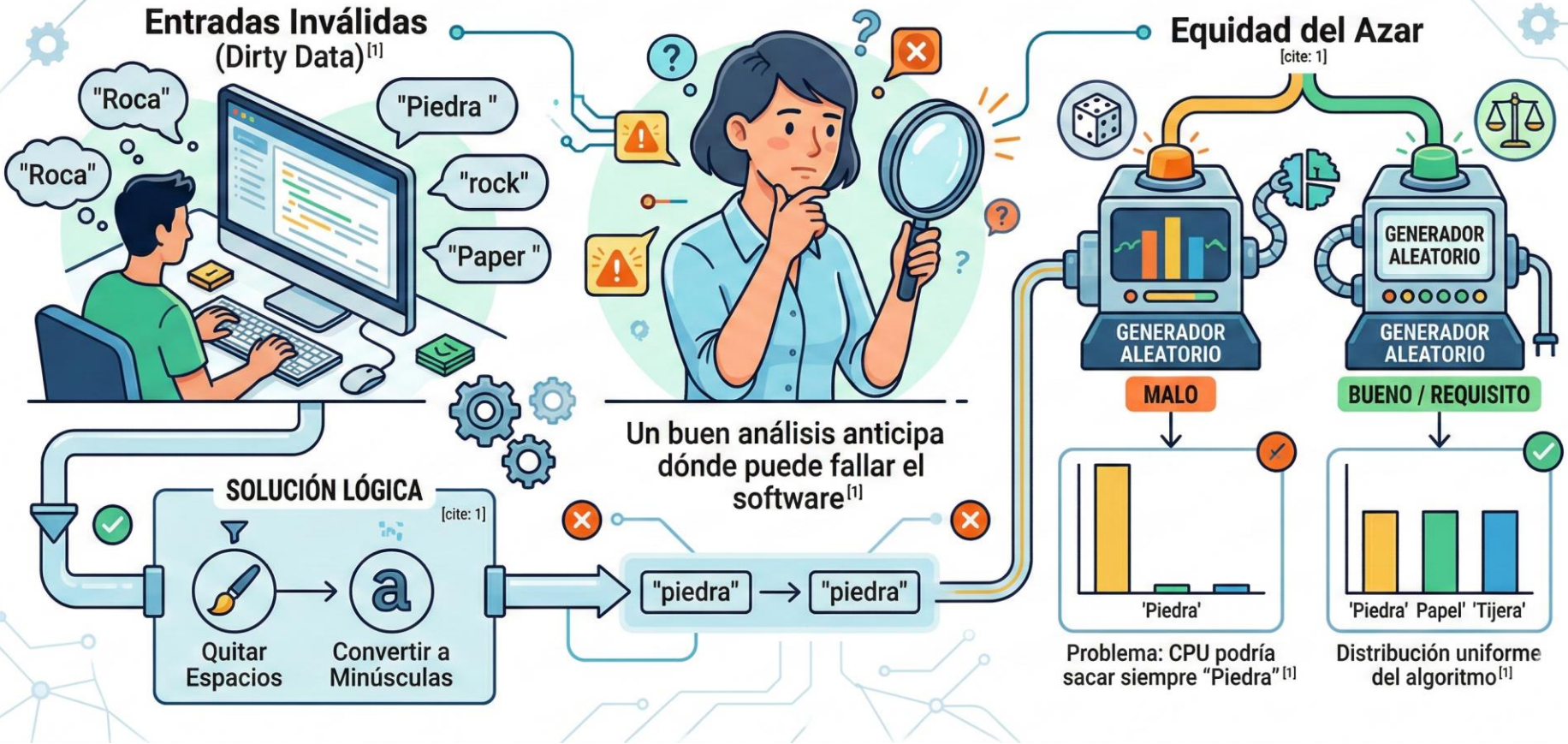
Solución Lógica: El sistema debe limpiar la entrada (quitar espacios, convertir a minúsculas) o rechazarla antes de jugar.

Equidad del Azar:

Problema: Si el generador aleatorio no es bueno, la CPU podría sacar siempre "Piedra".

Requisito: Asegurar que el algoritmo de aleatoriedad tenga una distribución uniforme.

PASO 4: IDENTIFICACIÓN DE RESTRICCIONES Y CASOS BORDE [cite: 1]



Diseño de Funcionalidades

1. **Identificación de Actores Antes de dibujar, definimos quién interactúa con el sistema.**

Actor Principal: Jugador (Usuario) - Quien inicia la acción.

Actor Secundario/Sistema: CPU (IA Simple) - El componente que responde y genera la jugada aleatoria.

1. Identificación de Actores [cite: 1]

Ambigüedad

[cite: 1]

Quien inicia la acción [cite: 1]

El componente que responde y genera la jugada aleatoria [cite: 1]

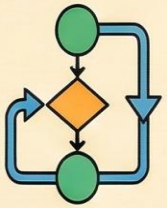
Actor Secundario/Sistema: **CPU** (IA Simple)

Actor Principal: **Jugador (Usuario)** [cite: 1]

Actor Secundario/Sistema: **CPU (IA Simple)** [cite: 1]

ALCANCE (SCOPE) [cite: 1]

Single jugador/ competitión 1v1 [cite: 1]



sola ronda/bucle [cite: 1]

No apuestas [cite: 1]

Multijugador [cite: 1]

2. Diseño del Diagrama de Casos de Uso (UML) Este diagrama define las "burbujas" de funcionalidad que tendrá tu software.

Descripción visual del diseño: Imagina una caja rectangular que representa los límites de tu sistema ("La Aplicación").
Fuera de ella está el "Jugador". Dentro, tenemos las siguientes funciones (Casos de Uso):

Iniciar Partida: El disparador del juego.

Seleccionar Movimiento: La acción principal del jugador (elegir Piedra, Papel o Tijera).

Validar Selección: Una función interna (marcada con <>) que se asegura de que el usuario no ingresó basura.

Generar Movimiento CPU: Función interna donde el sistema actúa. Calcular Ganador: El motor lógico que compara las dos elecciones. Mostrar Resultado: La interfaz final que comunica quién ganó.

Raptor - Proyecto1_Piedra_papel_tijera_v1.rap

File Edit Scale View Run Mode Ink Window Generate Help

125%

Symbols

Assignment

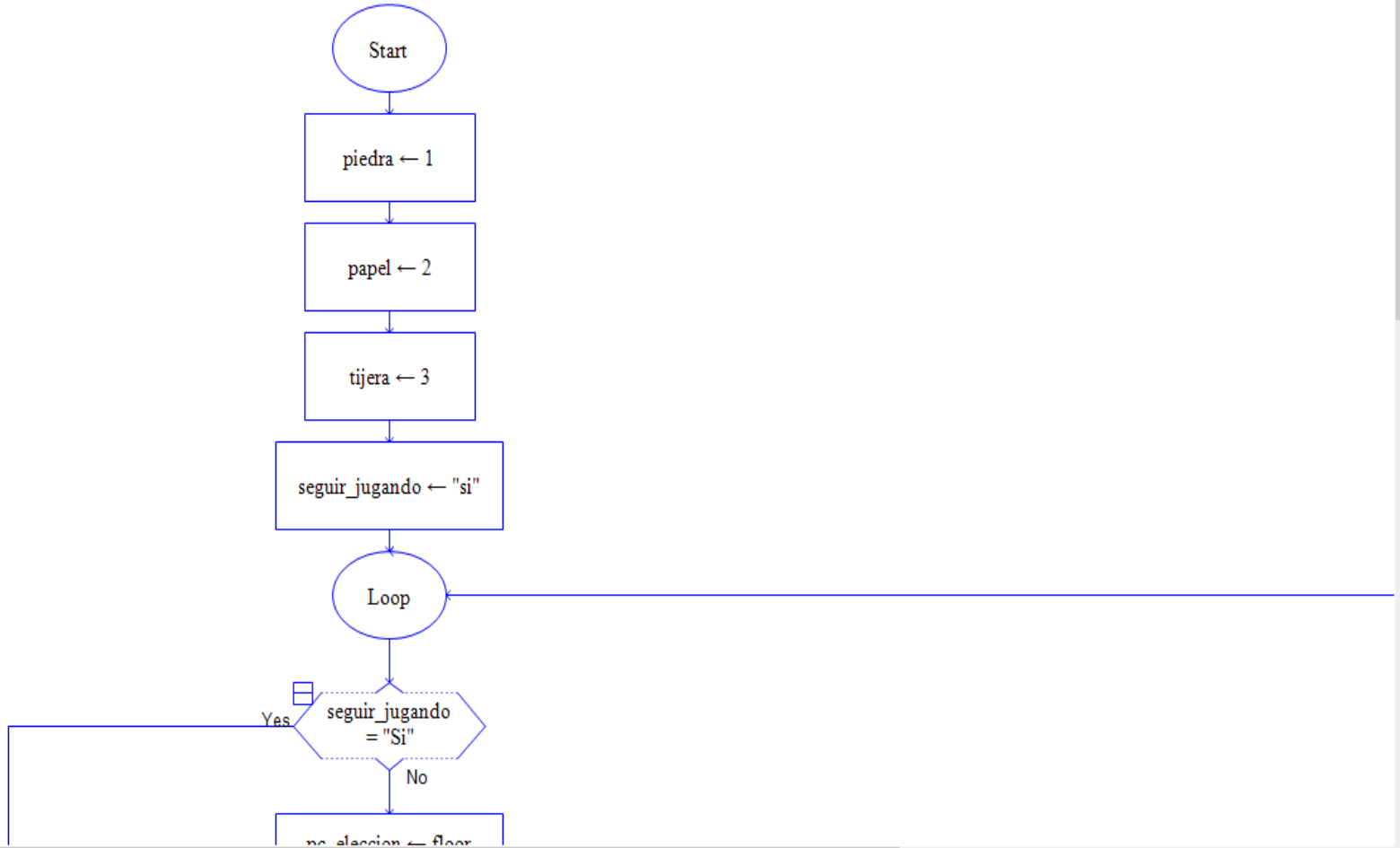
Call

Input Output

Selection

Loop

main



Buscar

ESP 11:32 AM 3/1/2026

Raptor - Proyecto1_Piedra_papel_tijera_v1.rap

File Edit Scale View Run Mode Ink Window Generate Help

80%

Symbols

Assignment

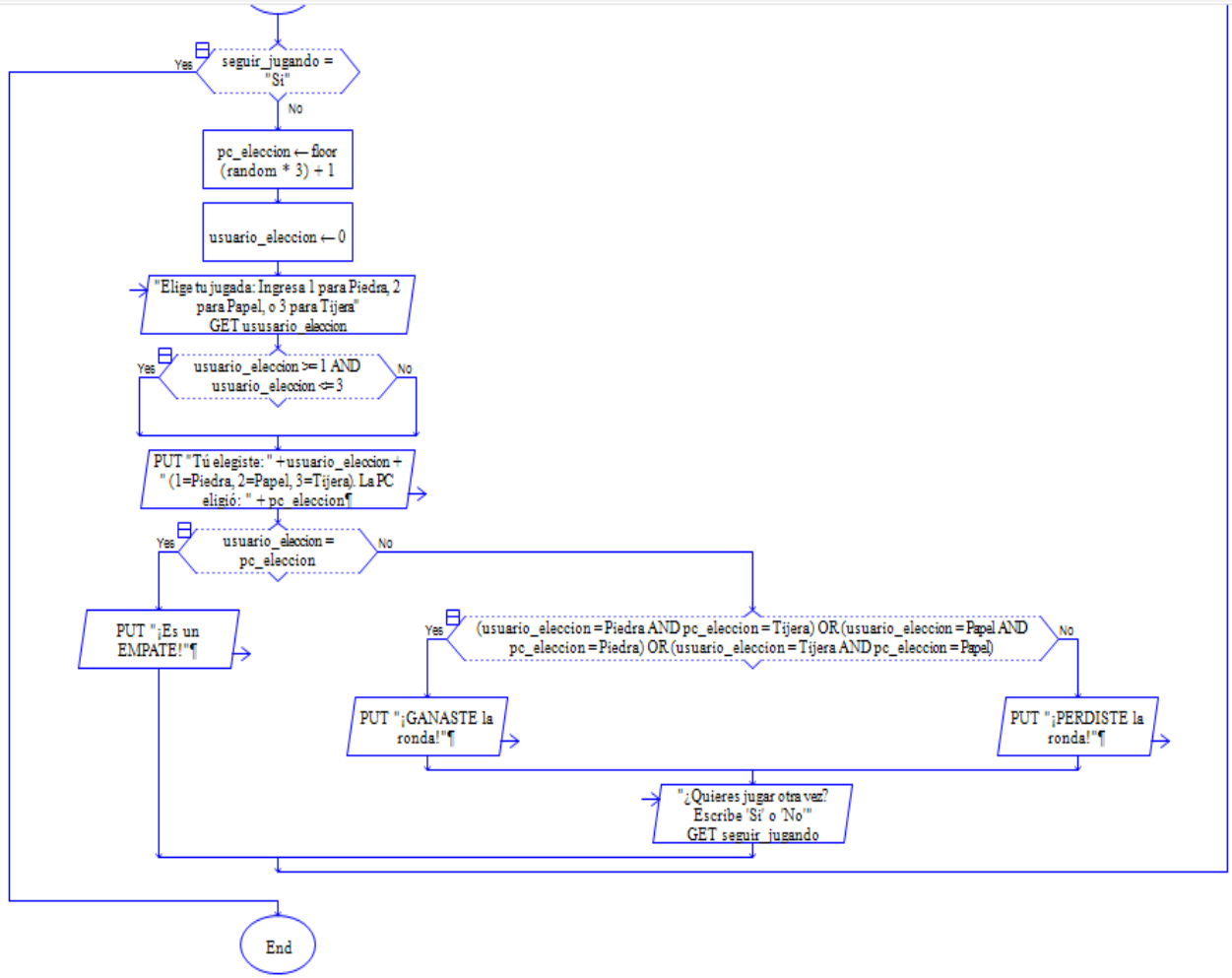
Call

Input Output

Selection

Loop

main



Buscar

ESP 11:34 AM 3/1/2026

3. Especificación Detallada de los Casos de Uso Un diagrama son solo dibujos si no tiene una narrativa. Aquí diseñamos el "contrato" de comportamiento para la funcionalidad principal: "Jugar una Ronda". ID Caso de UsoCU-01: Jugar Ronda Actor Jugador Precondición. El programa debe estar iniciado y listo para recibir input. Flujo Principal (Camino Feliz)

1. El Sistema solicita al Jugador que elija una opción.
2. El Jugador ingresa su elección (Piedra, Papel, Tijera).
3. El Sistema Valida la entrada (ver Flujo Alternativo A).
4. El Sistema ejecuta Generar Movimiento CPU (azar).
5. El Sistema Calcula el Ganador comparando ambas elecciones.
6. El Sistema muestra: Elección CPU + Mensaje de Victoria/Derrota.

Flujo Alternativo A (Error)

3a. El Jugador ingresa un texto inválido (ej: "Fuego").
3b. El Sistema muestra mensaje: "Opción no válida".
3c. El Sistema vuelve al paso

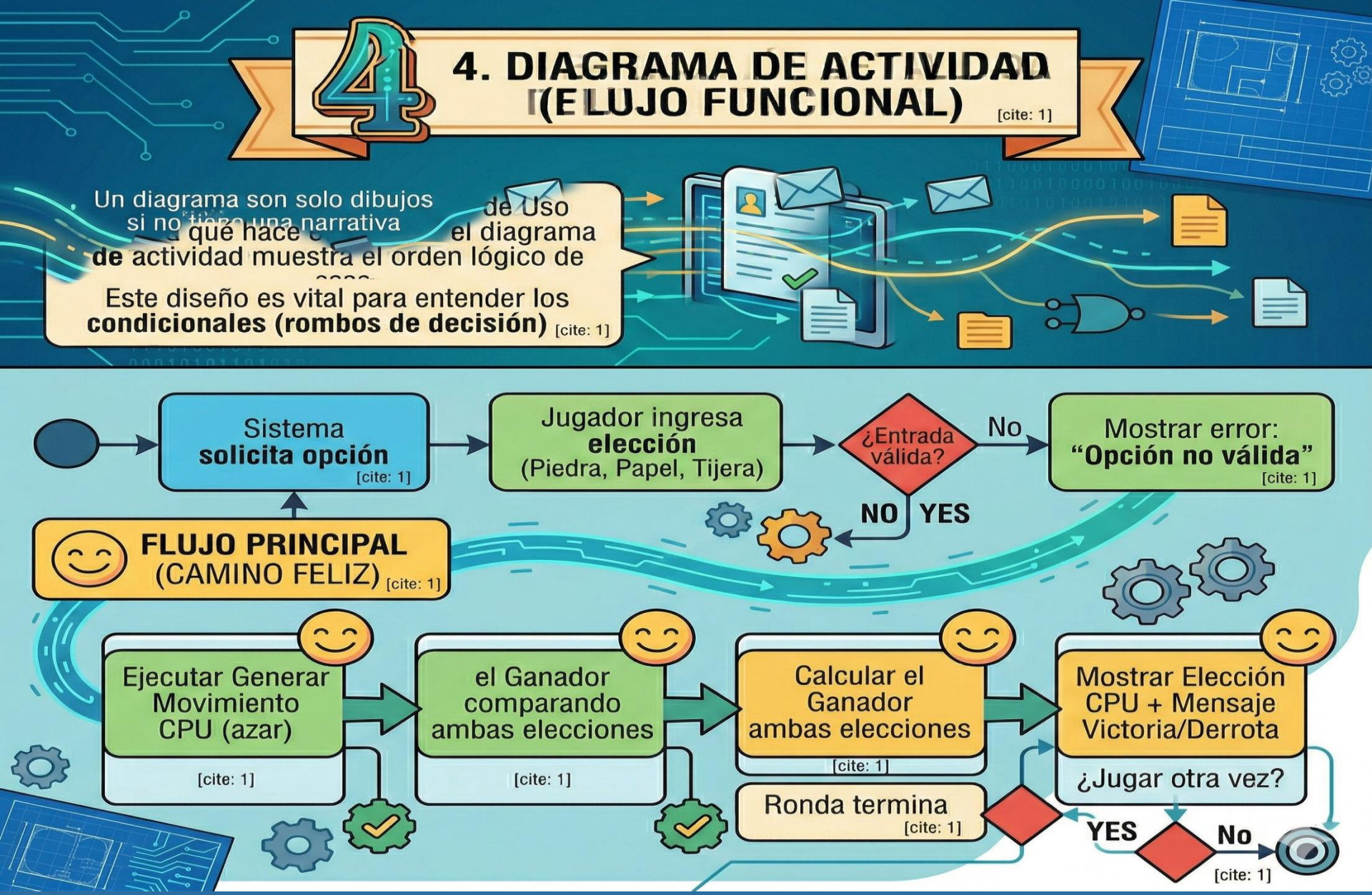
1. juego termina la ronda y pregunta si se desea jugar otra vez.

2. Diagrama de Actividad (Flujo Funcional) Mientras el diagrama de Casos de Uso muestra qué hace el sistema, el diagrama de actividad muestra el orden lógico de esas funciones. Este diseño es vital para entender los condicionales (rombos de decisión)

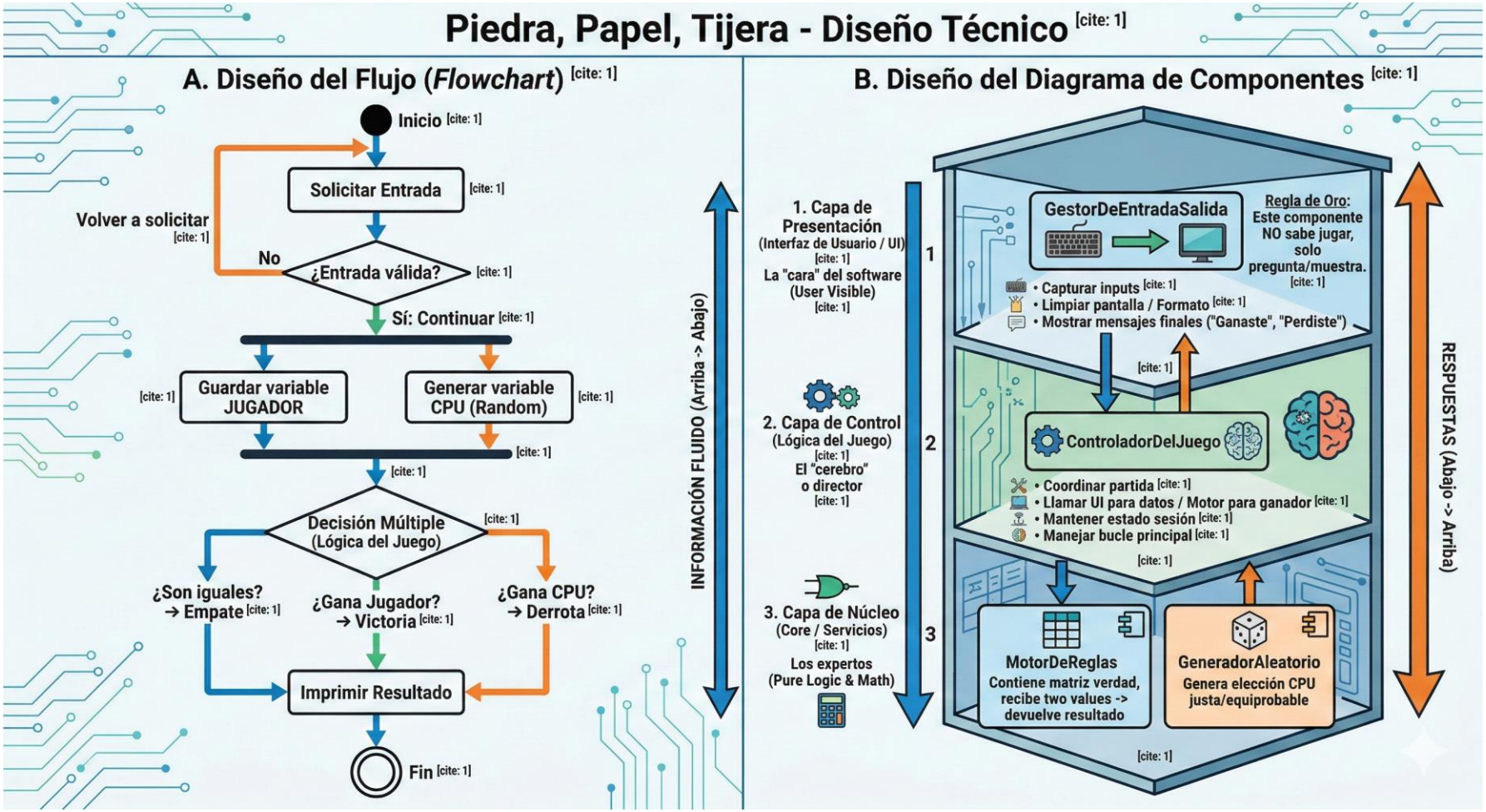
4. DIAGRAMA DE ACTIVIDAD (EL FLUJO FUNCIONAL) [cite: 1]

Un diagrama son solo dibujos si no tiene una narrativa de Uso qué hace el diagrama de actividad muestra el orden lógico de

Este diseño es vital para entender los **condicionales (rombos de decisión)** [cite: 1]



Diseño del Flujo



Elige tu jugada

Piedra

Papel

Tijera