VICERRECTORADO DE INVESTIGACIÓN INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DE SOFTWARE
SEGUNDA PROMOCIÓN

Modelo Neuronal de Estimación para el Esfuerzo de Desarrollo en Proyectos de Software (MONEPS)

ING. MARIO GIOVANNY ALMACHE CUEVA
ING. JENNY ALEXANDRA RUIZ ROBALINO

Julio 2015

AGENDA

1. Antecedentes
2. Trabajos relacionados
3. Modelo neuronal propuesto
4. Resultados obtenidos
5. Conclusiones y Recomendaciones

Necesidad de mejorar la precisión en la estimación del esfuerzo

- ✓ Los errores en la estimación de esfuerzo son todavía demasiado grandes.
- ✓ Se hace una mínima consideración de los aspectos no funcionales del software.
- ✓ No hay un claro entendimiento sobre las relaciones causales entre factores y los resultados finales de la estimación de esfuerzo en software.

Algunos enfoques para estimar el esfuerzo

✓Categorías fundamentales:

- Modelos Algorítmicos
- Juicio del Experto
- Machine Learning

✓Algunos modelos algorítmicos:

- Puntos de Función (PF)
- Líneas de Código (LDC)
- Puntos de Caso de Uso (PCU)

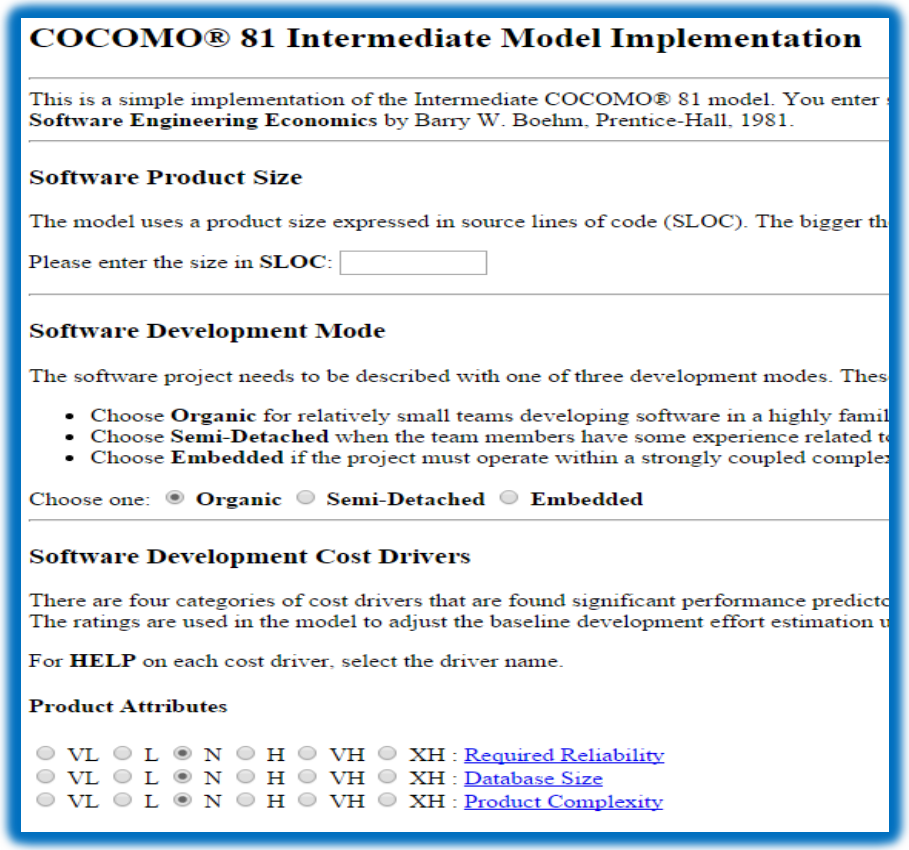
Modelos algorítmicos Vs. Juicio del experto

Enfoque	Ventajas	Inconvenientes	Aplicación idónea
Modelos algorítmicos	• Entradas y parámetros concretos. • Objetividad. • Eficiencia en cálculos.	• No prestan atención a circunstancias excepcionales. • Rechazan opiniones subjetivas.	• Proyectos con escasas alteraciones accidentales, con equipos de desarrollo estables y productos sencillos.
Juicio del experto	• Gran cantidad de opiniones subjetivas. • Consideración de circunstancias excepcionales.	• Dependencia de los expertos. • Posturas de expertos difíciles de adoptar.	• Primeras fases de desarrollo del producto.

Modelos algorítmicos Vs. Juicio del experto

Enfoque	Ventajas	Inconvenientes	Aplicación idónea
Modelos algorítmicos	• Entradas y parámetros concretos. • Objetividad. • Eficiencia en cálculos.	• No prestan atención a circunstancias excepcionales. • Rechazan opiniones subjetivas.	• Proyectos con escasas alteraciones accidentales, con equipos de desarrollo estables y productos sencillos.
Juicio del experto	• Gran cantidad de opiniones subjetivas. • Consideración de circunstancias excepcionales.	• Dependencia de los expertos. • Posturas de expertos difíciles de adoptar.	• Primeras fases de desarrollo del producto.

Dos modelos algorítmicos: Cocomo 81 y Cocomo II



COCOMO® 81 Intermediate Model Implementation

This is a simple implementation of the Intermediate COCOMO® 81 model. You enter
Software Engineering Economics by Barry W. Boehm, Prentice-Hall, 1981.

Software Product Size

The model uses a product size expressed in source lines of code (SLOC). The bigger th

Please enter the size in **SLOC**:

Software Development Mode

The software project needs to be described with one of three development modes. Thes

- Choose **Organic** for relatively small teams developing software in a highly famil
- Choose **Semi-Detached** when the team members have some experience related t
- Choose **Embedded** if the project must operate within a strongly coupled comple

Choose one: **Organic** **Semi-Detached** **Embedded**

Software Development Cost Drivers

There are four categories of cost drivers that are found significant performance predict
The ratings are used in the model to adjust the baseline development effort estimation u

For **HELP** on each cost driver, select the driver name.

Product Attributes

VL L N H VH XH : Required Reliability
VL L N H VH XH : Database Size
VL L N H VH XH : Product Complexity

Figura No. 1: Cocomo 81 Calculator

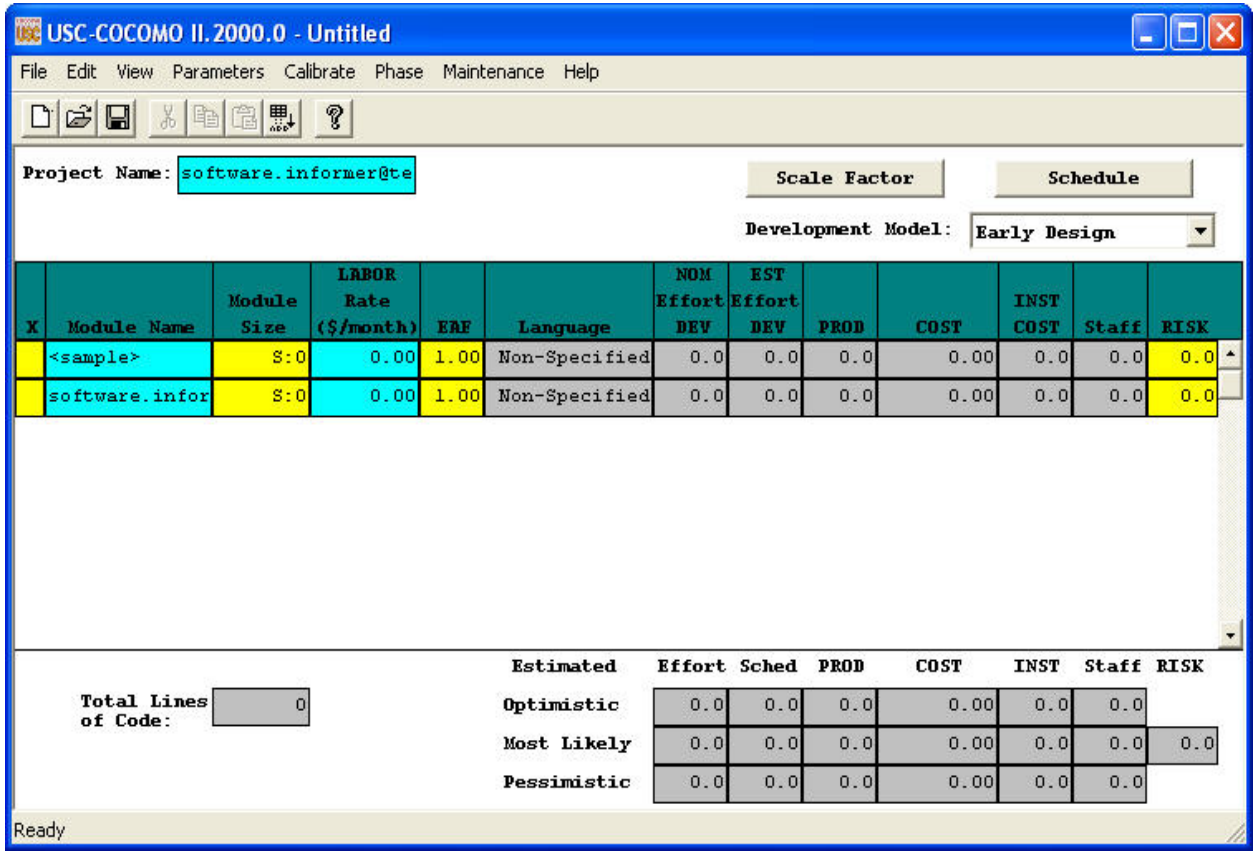
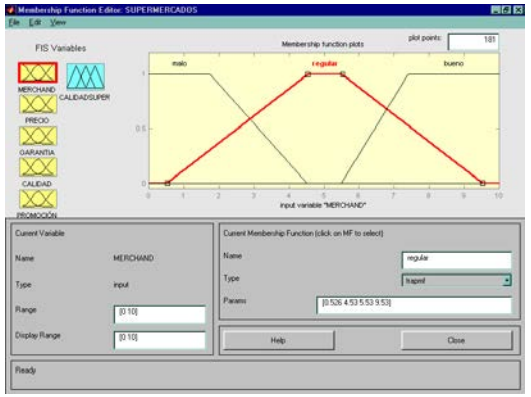


Figura No. 2: USC Cocomo II

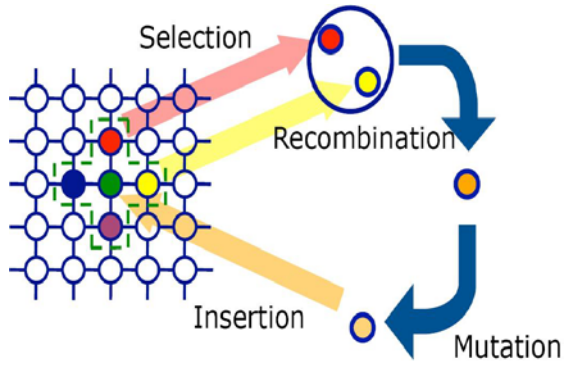
Machine Learning (Aprendizaje Automático)

- ✓ Lógica difusa

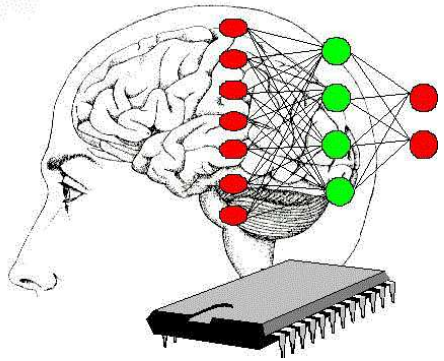


- ✓ Minería de datos

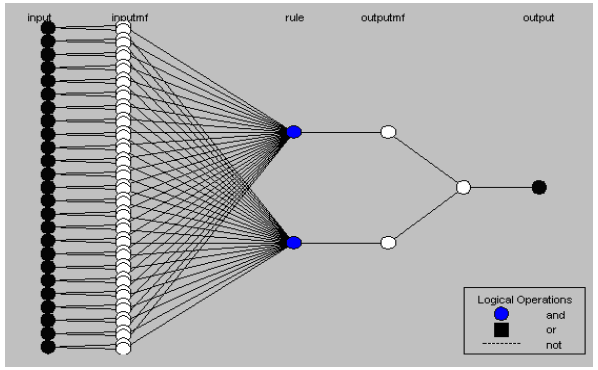
- ✓ Algoritmos genéticos



- ✓ Redes neuronales artificiales



- ✓ Sistemas neuro-difusos



Redes Neuronales Artificiales en Backpropagation

Características importantes:

- ✓ Aprenden de manera supervisada e inductiva.
- ✓ Son suficientes 3 capas para las tareas de aprendizaje e identificación de patrones.
- ✓ Poca complejidad estructural y algorítmica.
- ✓ Buena disponibilidad de herramientas automatizadas (libres/propietarias) para su implementación.

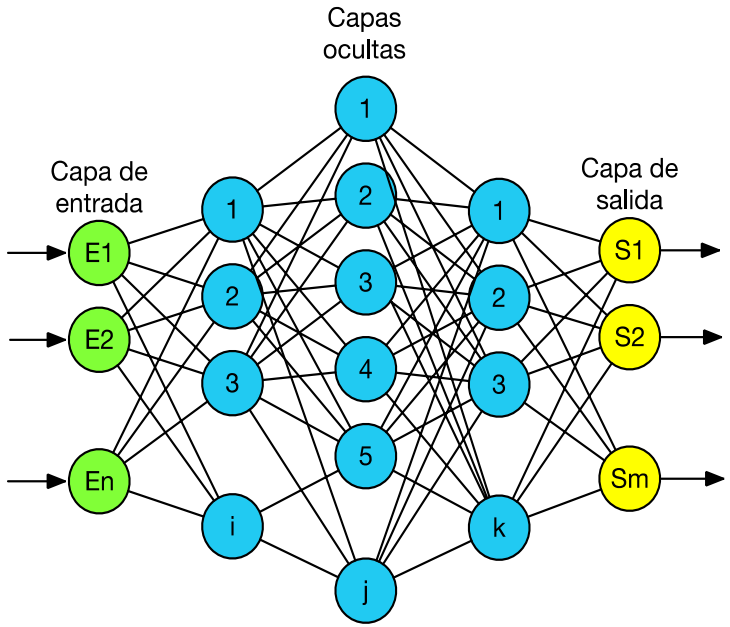


Figura No. 3: Ejemplo de RNA con cuatro capas

Aplicaciones de las RNA

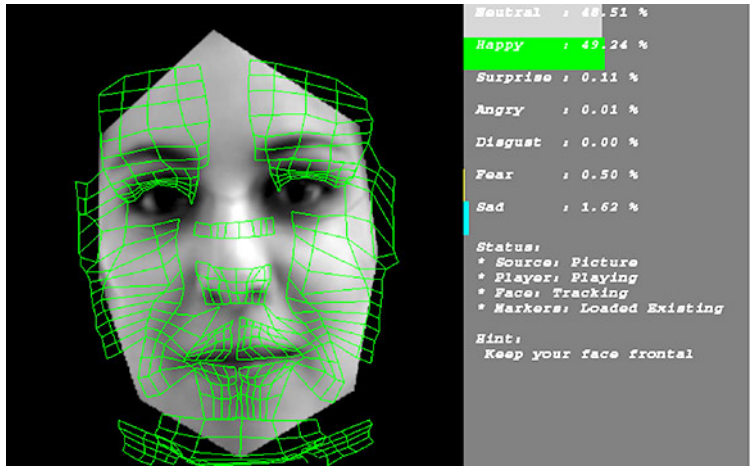


Figura No. 4:
Reconocimiento Facial

Figura No. 5: Procesamiento de imágenes

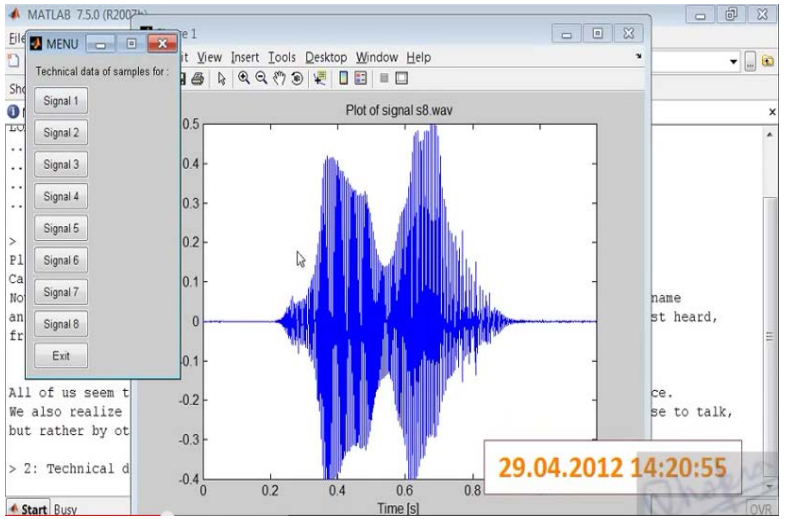
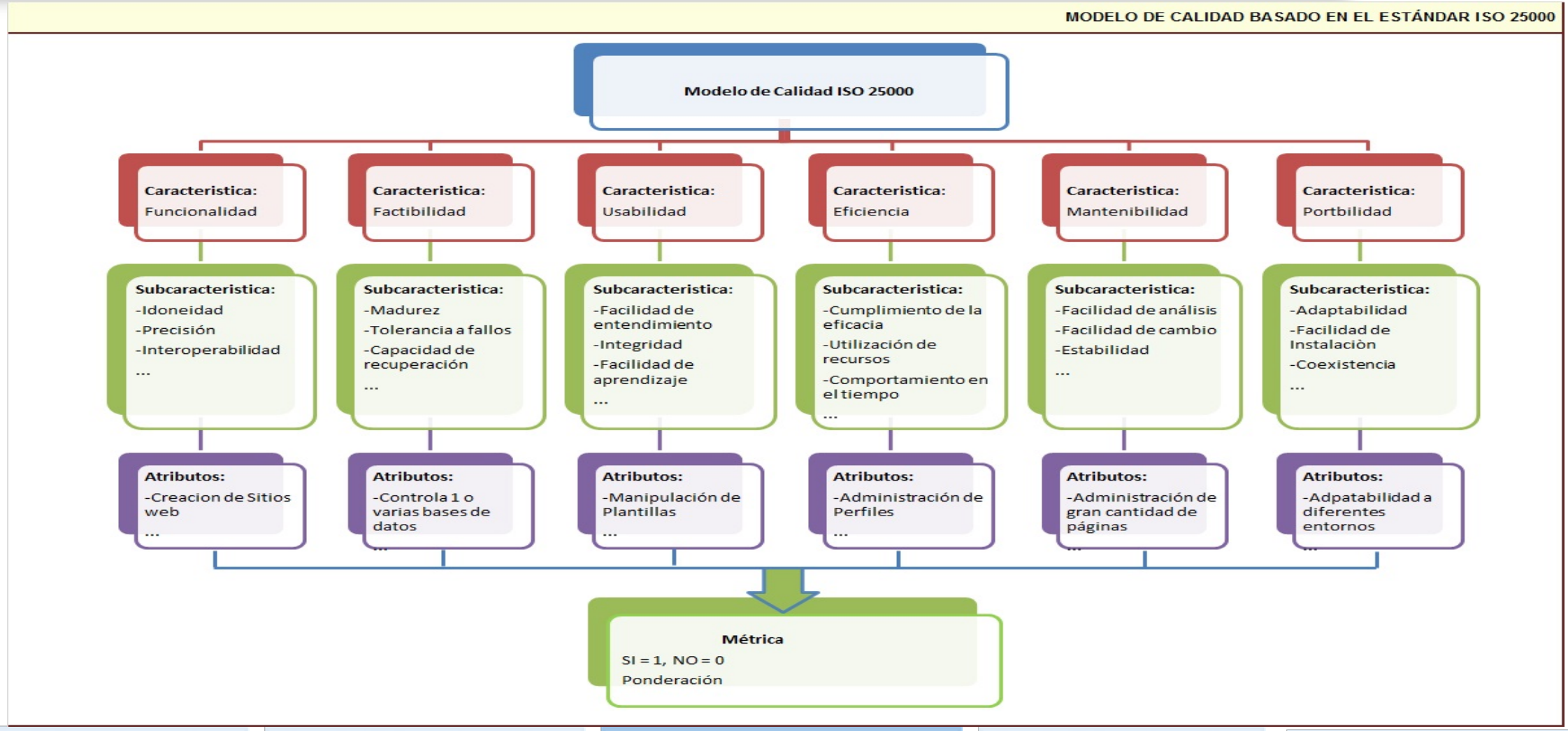


Figura No. 6:
Reconocimiento de voz

Modelo Neuronal Propuesto: Modelo de Calidad ISO 25000



Codificación de Atributos en MONEPS

Tabla No. 1: Algunos atributos tomados de la norma ISO 25000

Atributo	Código	Valores	Descripción
Nivel de seguridad	A2	Alto, Medio, Bajo	• Indica el nivel de seguridad requerido para la aplicación.
Número de programadores	M3	1, 2, 3, ...	• Número de integrantes del equipo de desarrollo asignados.
Experiencia del equipo de desarrollo	M1	Alta, Media, Baja	• Indica la experiencia del equipo de desarrollo en aplicaciones similares.
Lenguaje de Programación	F1	Imperativo, Declarativo, Orientado a Objetos, Orientado al Problema	• Tipo de lenguaje de programación utilizado.
Complejidad del sistema	I4	Alta, Media, Baja	• Complejidad prevista para el sistema.

Modelo Neuronal Propuesto

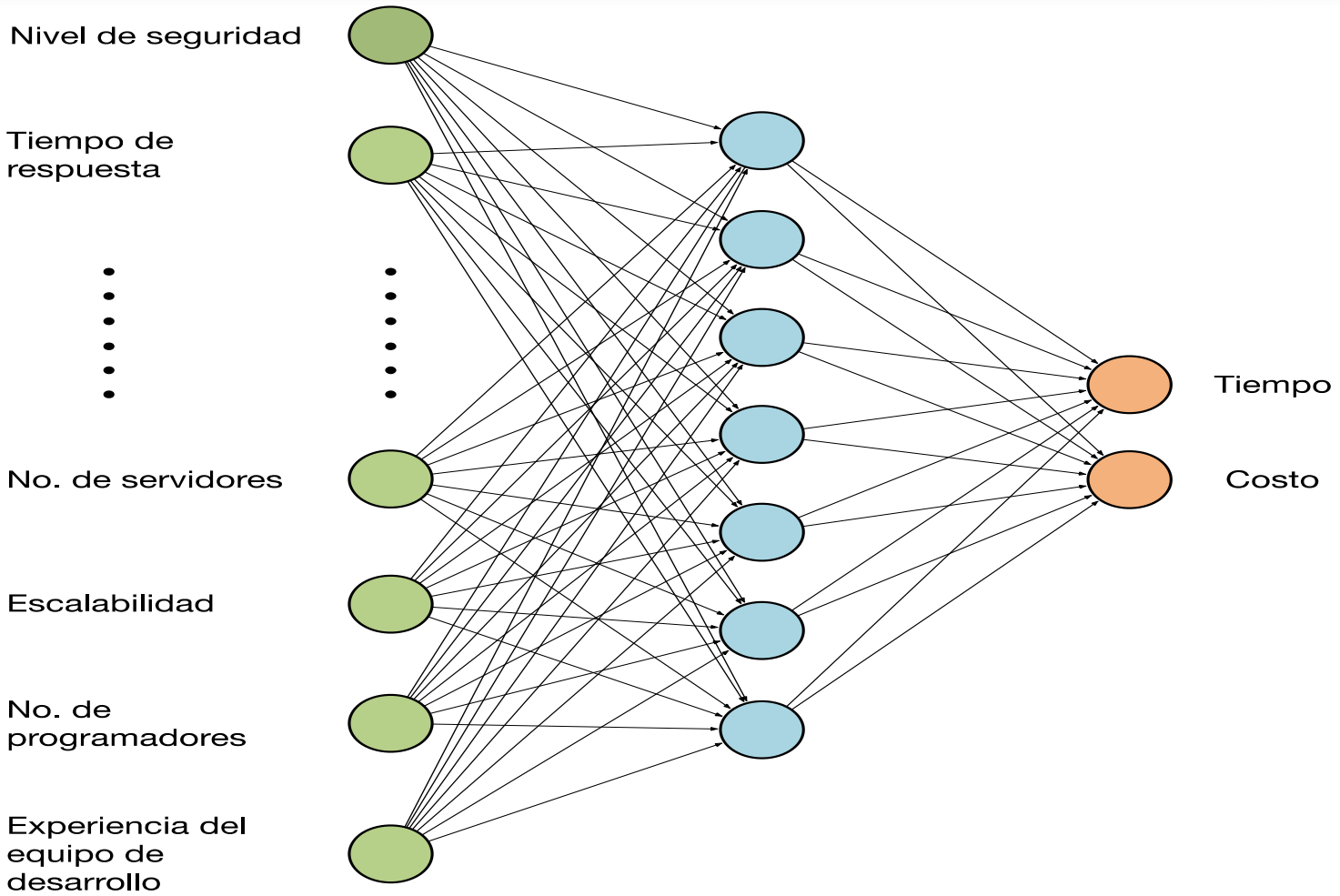
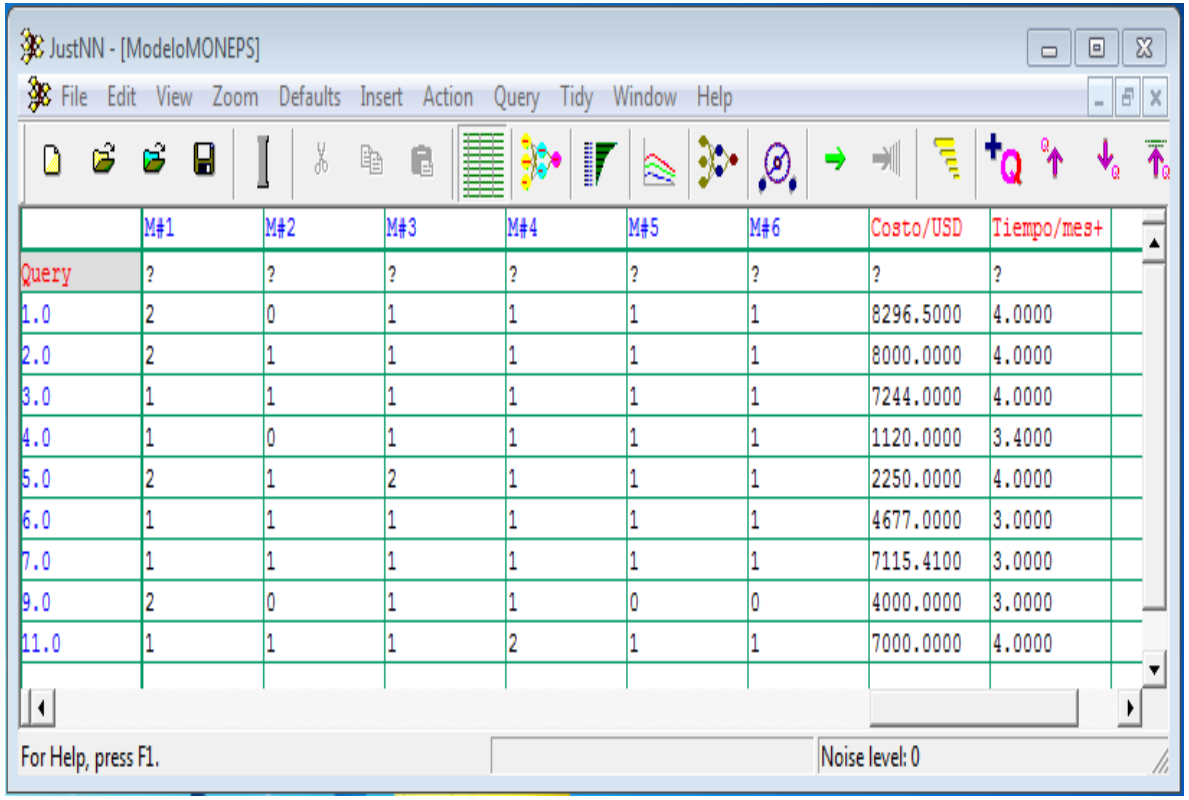


Figura No. 7: RNA simplificada y utilizada en Moneps

Resultados obtenidos: herramienta JustNN



	M#1	M#2	M#3	M#4	M#5	M#6	Costo/USD	Tiempo/mes+
Query	?	?	?	?	?	?	?	?
1.0	2	0	1	1	1	1	8296.5000	4.0000
2.0	2	1	1	1	1	1	8000.0000	4.0000
3.0	1	1	1	1	1	1	7244.0000	4.0000
4.0	1	0	1	1	1	1	1120.0000	3.4000
5.0	2	1	2	1	1	1	2250.0000	4.0000
6.0	1	1	1	1	1	1	4677.0000	3.0000
7.0	1	1	1	1	1	1	7115.4100	3.0000
9.0	2	0	1	1	0	0	4000.0000	3.0000
11.0	1	1	1	2	1	1	7000.0000	4.0000

Figura No. 8: Carga de datos en JustNN

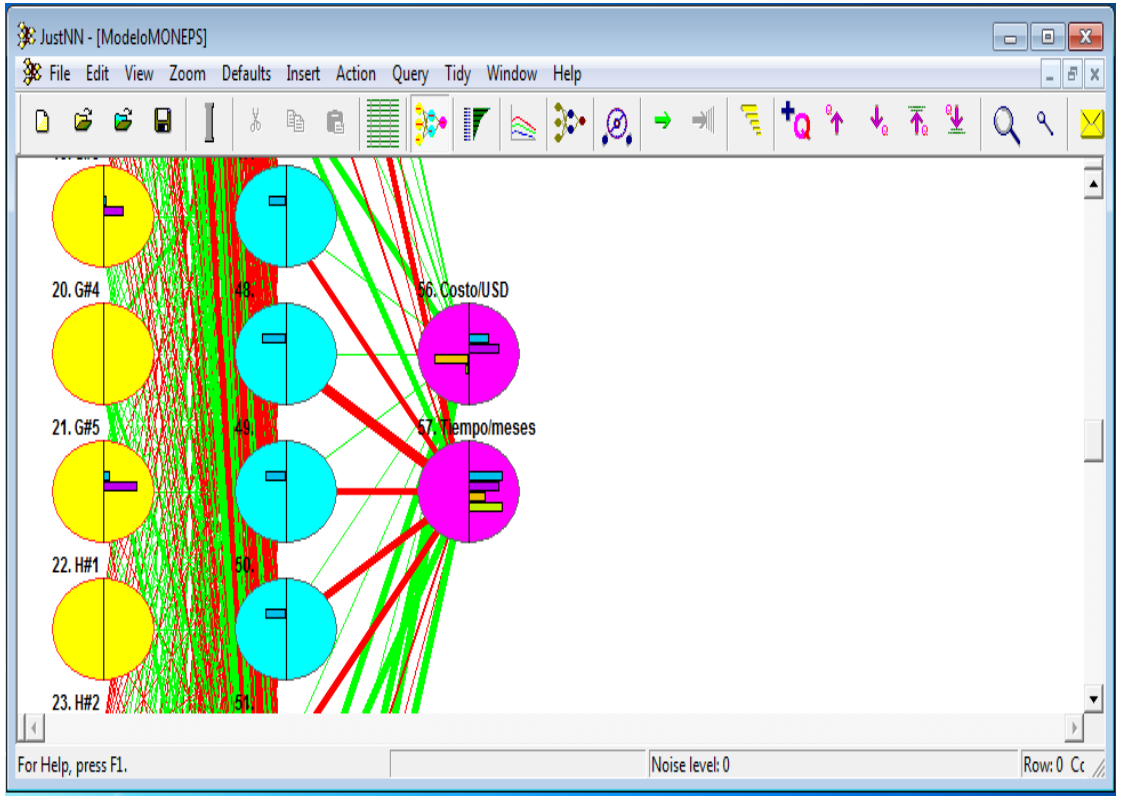


Figura No. 9: Red neuronal artificial en JustNN

Entrenamiento de la RNA usada por MONEPS

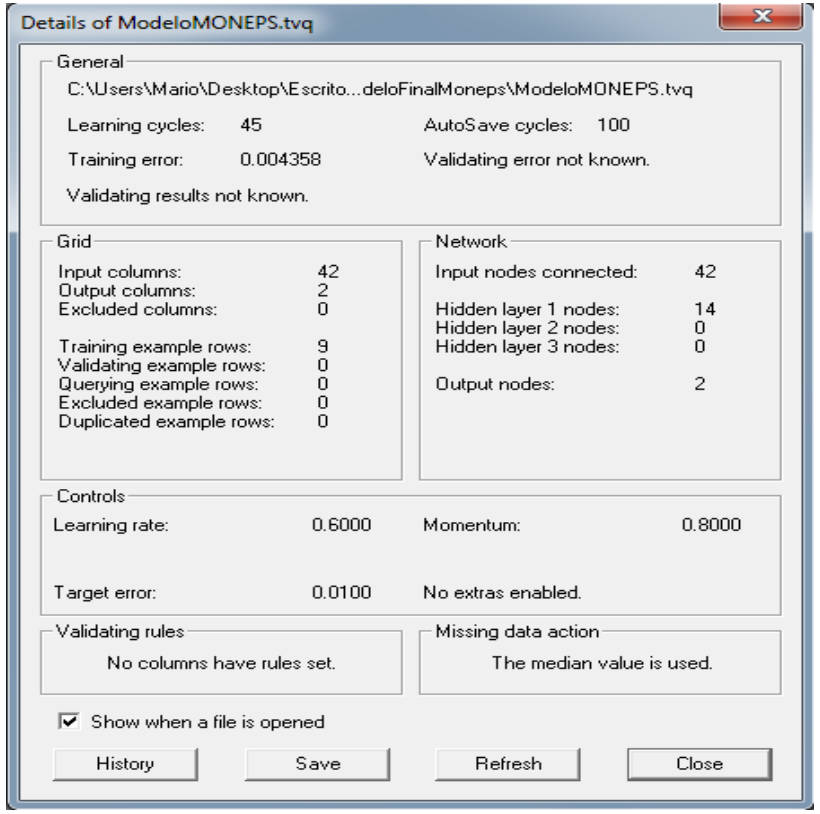


Figura No. 10: Resumen de entrenamiento para la RNA

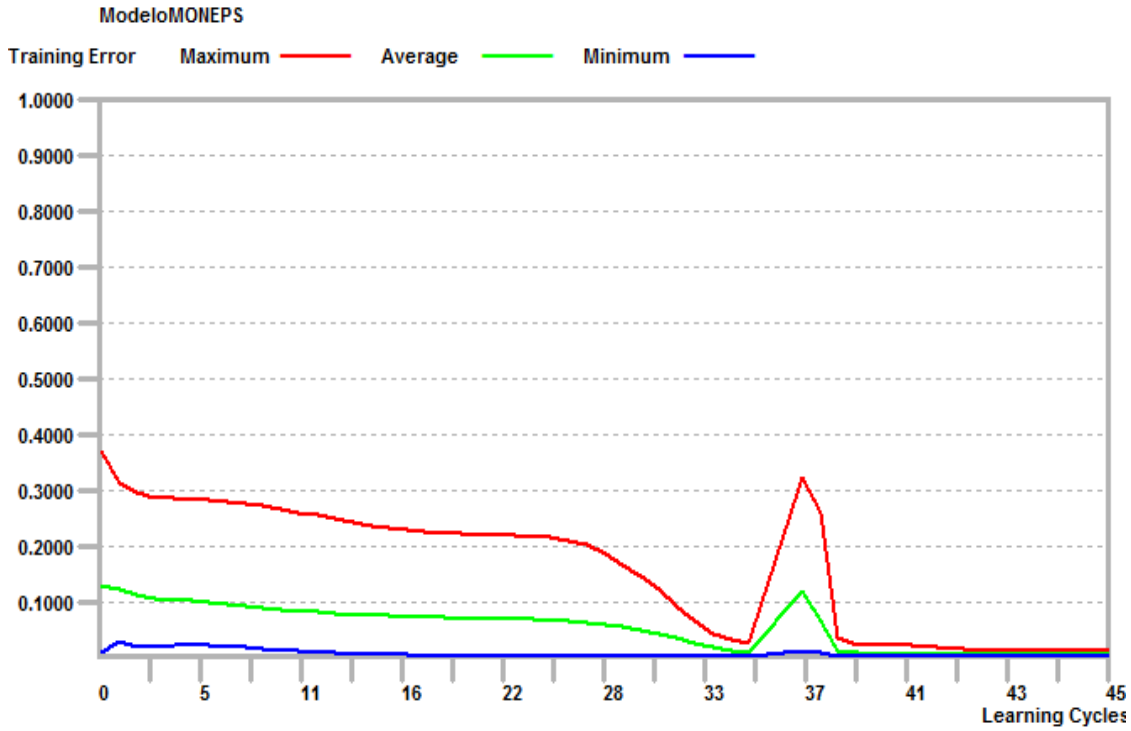


Figura No. 11: Error de la RNA durante la fase de entrenamiento

Estimación para 3 casos de prueba y contrastación con Cocomo 81

Tabla No. 2: Tiempo y costo estimados por Moneps para 3 proyectos de SW

Caso	Tiempo real de duración (meses)	Tiempo estimado (MONEPS)	Costo referencial (USD)	Costo estimado (MONEPS)
1	3.00	3.65	7558.80	8139.45
2	5.00	3.97	8810.00	7273.05
3*	4.00	3.96	7244.00	7797.52

Tabla No. 3: Cocomo 81 Vs. Moneps

Caso	Tiempo (meses)		
	Real	Estimado por Cocomo 81	Estimado por Moneps
1	3.0	6.85	3.65
2	5.0	7.75	3.97

1. CODESOFT: Simulador para la evaluación de aptitudes de aspirantes para el desarrollo de software.
2. SIFFAAR: Sistema para automatizar el proceso de control de inventario y facturación en la empresa FAAR.
3. SICFO: Sistema para gestión de fichas odontológicas (parte del conjunto de entrenamiento)

Contrastación con Cocomo II

Caso	Nombre del proyecto	Tiempo (meses)			Costo (USD)		
		Real	Estimado por Cocomo-II	Estimado por Moneps	Referen.	Estimado por Cocomo-II	Estimado por Moneps
1	CODESOFT	3.0	12.20	3.65	7558.80	10371.01	8139.45
2	SIFFAAR	5.0	10.20	3.97	8810.00	12376.34	7273.05

Tabla No. 4: Cocomo II Vs. Moneps

Caso	Nombre del proyecto	Error relativo para el tiempo		Error relativo para el costo	
		Cocomo-II	Moneps	Cocomo-II	Moneps
1	CODESOFT	306.67%	21.67%	37.20%	7.68%
2	SIFFAAR	104.00%	20.60%	40.48%	17.45%

Tabla No. 5: Error relativo en Cocomo II y Moneps

Conclusiones:

- ✓ Se ha logrado construir una red neuronal en backpropagation cuyas entradas se fundamentan en los atributos del estándar para calidad de software ISO 25000, para estimar el costo y tiempo de desarrollo en productos software.

- ✓ La arquitectura de la red neuronal fue identificada en base a las características contemporáneas del software y, considerando el desempeño de las topologías neuronales disponibles.

- ✓ MONEPS ha logrado la convergencia de aspectos funcionales y no funcionales en la estimación de tiempo y costo para productos software.

Conclusiones:

- ✓ La identificación de métricas más especializadas para los aspectos no funcionales de un sistema, es un verdadero desafío en la Ingeniería del Software.
- ✓ Al menos, para proyectos académicos, MONEPS muestra estimaciones de tiempo y costo más cercanos a los reales que los modelos COCOMO 81 y COCOMO II.
- ✓ MONEPS es de fácil uso y escalable; permitiendo realizar los ajustes necesarios para mejorar el nivel de adecuación y precisión en la naturaleza dinámica del software.

Recomendaciones:

- Profundizar en el análisis de la familia de estándares ISO-25000 u otras normas orientadas a verificar la calidad del software, para un mejor entendimiento de los aspectos funcionales y no funcionales más críticos que inciden en el desarrollo de software.

- Para una mejor adaptabilidad de MONEPS, cada empresa/usuario puede configurar los atributos de entrada; así, el modelo neuronal podrá ajustarse a las restricciones de configuración impuestas.

- Otra mejora posterior de MONEPS se basa en la adición o eliminación de atributos, así como en la depuración de métricas para los aspectos no funcionales.

Recomendaciones:

- Para tener estimaciones de costo y tiempo, acorde a los requerimientos locales de software, es recomendable recopilar más casos de productos desarrollados en la empresa pública y/o privada, que permitan alimentar el modelo neuronal.

- Las empresas desarrolladoras de software deberían iniciar los proyectos realizando algún tipo de estimación temprana del esfuerzo requerido, lo que permitirá reducir riesgos de incumplimiento o pérdidas importantes de capitales.

- Se debe incentivar el estudio de métodos y técnicas alternativas para la estimación del esfuerzo en proyectos de software. No hay un modelo universal de estimación, y los disponibles tienen falencias contextuales que no han sido corregidas.

En 1899, Max Planck preguntó lo siguiente:

¿cómo sería posible crear un sistema de unidades objetivo y universal?

¿La idea de Planck es extensible al Software?

GRACIAS POR SU ATENCIÓN