Capítulo 6: Introducción a los Sistemas Basados en el Conocimiento (SBC)

6.1 Introducción

En Inteligencia Artificial (IA), la resolución de problemas es esencial. Los métodos generales iniciales usaban funciones heurísticas básicas, sin conocimiento específico del dominio, lo que resultaba ineficiente para problemas reales debido a su alto coste computacional.

Para mejorar la eficiencia, se introdujeron los métodos fuertes, que utilizan conocimiento específico del dominio. Estos sistemas requieren mayor esfuerzo de desarrollo y no son fácilmente reutilizables.

Sistemas Expertos vs. SBC

- Sistemas Expertos: Basados en reglas de expertos humanos, cerrados, sin foco en aprendizaje.
- SBC (Sistemas Basados en Conocimiento): Más generales, integran diversas fuentes de conocimiento y metodologías (reglas, casos, redes neuronales), permiten aprendizaje y adaptación.

6.2 Características de los SBC

Los SBC resuelven problemas complejos donde el software convencional no es aplicable. Deben poseer:

- Flexibilidad para enfrentar diferentes tipos de problemas.
- Capacidad de emular razonamiento humano (soluciones + justificación).
- Trabajo con gran volumen de información, con incertidumbre e incompletitud.
- Uso de información simbólica y lenguaje natural.
- Capacidad de aprender y adaptarse.

Diseño de SBC – Aspectos Clave

1. Separación entre conocimiento y control: Facilita modularidad, extensibilidad, modificabilidad e independencia.
2. Conocimiento heurístico: Basado en experiencia humana, no algorítmico.
3. Interactividad: Comunicación con expertos, otros sistemas y entornos; justificación de decisiones.

Áreas integradas de la IA en SBC

- Representación del conocimiento
- Razonamiento e inferencia (incluso con incertidumbre)
- Búsqueda heurística
- Interacción en lenguaje natural
- Aprendizaje (adquisición automática de conocimiento)

6.3 Necesidad de los SBC

Surge por el alto coste y escasez de expertos humanos. Los SBC permiten:

- Disponibilidad del conocimiento experto de forma continua.
- Asistencia a expertos y formación de nuevos.
- Preservar y combinar conocimientos de múltiples expertos.
- Soluciones rápidas y justificadas.
- Manejo de grandes volúmenes de datos.
- Decisiones autónomas en ciertos contextos.

6.4 Problemas adecuados para SBC

No cualquier problema se puede o se debe resolver con un SBC. Su desarrollo implica costes elevados en tiempo, recursos humanos y económicos, por lo que antes de iniciar el proyecto es clave realizar un análisis previo. A continuación, se detallan los criterios esenciales que ayudan a determinar la viabilidad y conveniencia de desarrollar un SBC para un problema concreto:

1. ¿La necesidad de la solución justifica el desarrollo? Sí, si el problema es importante y complejo. Desarrollar un SBC implica un coste significativo, por lo que la problemática que se quiere resolver debe tener suficiente entidad o relevancia. Esto significa que debe tratarse de un problema cuya resolución aporte un valor importante y que no se pueda resolver de forma simple o económica por otros medios.
2. ¿Es posible acceder al experto necesario? Para construir un SBC es imprescindible contar con la colaboración de un experto que posea el conocimiento específico sobre el problema. Este experto debe estar disponible y dispuesto a compartir su conocimiento y colaborar activamente en el proceso de formalización, es decir, transformar lo que sabe en reglas y estructuras que puedan ser implementadas en el sistema.
   Problema común: A veces los expertos no están disponibles o les cuesta explicar cómo resuelven los problemas (esto se conoce como la paradoja del experto).
3. ¿El problema puede plantearse como un proceso de razonamiento simbólico? Un SBC utiliza razonamiento simbólico, es decir, opera con información representada mediante símbolos, conceptos y reglas lógicas, en lugar de simples cálculos numéricos.
   Requisito clave: Si no se puede formular el problema de esta manera (por ejemplo, si es puramente numérico o matemático), entonces no es adecuado para un SBC, y probablemente sea mejor usar métodos tradicionales como algoritmos o modelos matemáticos.
4. ¿El problema está bien estructurado? Para que el SBC funcione correctamente, se debe poder identificar y formalizar todos los elementos del problema: variables, relaciones, procedimientos, datos y reglas.
   Si el problema es vago o difícil de definir formalmente, será muy difícil construir un SBC útil. Es necesario que el problema tenga estructura lógica y componentes definibles para que el sistema pueda razonar sobre ellos.
5. ¿Puede resolverse el problema por métodos tradicionales? Antes de construir un SBC se debe preguntar: ¿Hay una solución más sencilla disponible?
   Si el problema tiene una solución algorítmica clara, directa y eficiente (por ejemplo, una fórmula matemática o un procedimiento estándar), entonces no es necesario construir un SBC, ya que estos sistemas están diseñados para casos donde el conocimiento no puede codificarse de forma simple y directa.
6. ¿Existen expertos disponibles y cooperativos? La construcción del SBC requiere colaboración constante con expertos durante todo el proceso. Estos expertos deben:

- Entender el objetivo del SBC.
- Estar dispuestos a formalizar su conocimiento (explicarlo, estructurarlo).
- Participar activamente en pruebas y validaciones.

Sin esta cooperación, el desarrollo del SBC no es viable, ya que el sistema se basa en el conocimiento que los expertos proporcionan.

7. ¿Está el problema bien dimensionado Un SBC debe enfocarse en un problema que sea lo suficientemente limitado y acotado como para poder desarrollarse, probarse y utilizarse eficazmente.
   Si el problema requiere una cantidad inmanejable de conocimiento, o es demasiado amplio o complejo, puede que no sea factible formalizarlo adecuadamente o completar el proyecto. El SBC debe diseñarse de forma que el conocimiento necesario sea manejable y alcanzable.

6.5 Limitaciones y Problemas de los SBC
- Fragilidad: Fuera de su dominio, no ofrecen respuestas.
- Control del razonamiento: Difícil de replicar el metaconocimiento humano.
- Baja reutilización de conocimientos entre dominios.
- Dificultad para integrar aprendizaje con conocimiento existente.
- Adquisición del conocimiento: Cuello de botella (paradoja del experto).
- Validación compleja: Riesgo de inconsistencias y alta carga de comprobación.

6.6 Áreas de Aplicación
Usados en múltiples áreas (medicina, biología, diseño, finanzas, meteorología), resuelven:
- Identificación: diagnóstico, supervisión, predicción.
- Construcción de soluciones: planificación, diseño, configuración.

6.7 Historia Breve de los SBC
- DENDRAL (1965): Identificación de moléculas. Primer sistema experto.
- MYCIN (1970): Diagnóstico médico, introdujo factores de certeza.
- PROSPECTOR (1974): Prospección minera, nuevo tratamiento de incertidumbre.
- XCON (1980): Configuración de sistemas VAX, éxito comercial.

Desde 1980, se multiplican los sistemas expertos. En los 90, se diversifican técnicas (agentes, casos, redes neuronales) y se adopta el término SBC. Se integran nuevas técnicas de incertidumbre (redes bayesianas, razonamiento difuso) y aprendizaje automático.

Capítulo 7: Arquitectura de los Sistemas Basados en el Conocimiento (SBC)

7.1 Introducción: Componentes esenciales de un SBC
Un SBC debe ser capaz de:
- Resolver problemas con información simbólica.
- Aplicar razonamiento heurístico.
- Explicar sus decisiones.
- Interactuar con el entorno o el usuario.

Para lograr esto, todo SBC incluye:
1. Almacenamiento del conocimiento (Base de Conocimiento).
2. Uso e interpretación del conocimiento (Motor de Inferencia).
3. Almacenamiento del estado del problema (Memoria de Trabajo).
4. Justificación e inspección de soluciones.
5. Interfaz de comunicación con el usuario/entorno.
6. (Opcional) Subsistema de aprendizaje.

7.2.1 Almacenamiento del Conocimiento
Contiene:
- Conocimiento factual (objetos y atributos).
- Conocimiento relacional (relaciones entre objetos).
- Conocimiento condicional (reglas de producción).

El conocimiento factual y relacional se organiza como una ontología, usando estructuras como frames o redes semánticas. Las reglas de producción pueden ser lógicas, tratar incertidumbre o tiempo, etc.
Tipos de conocimiento representado en reglas:
- Deductivo: cómo resolver problemas paso a paso.
- Sobre objetivos: estrategias generales.
- Causal: soporte para justificar deducciones.

Meta-reglas: reglas sobre el uso de otras reglas, organizadas en módulos para mejorar el control de resolución.

7.2.2 Uso e Interpretación del Conocimiento
Se realiza mediante un motor de inferencia que selecciona y aplica reglas para resolver el problema. Su eficacia depende de la complejidad de las reglas y la estrategia de resolución.

7.2.3 Almacenamiento del Estado del Problema
Llamada memoria de trabajo, guarda datos iniciales y deducciones intermedias. Puede incluir preferencias, caminos alternativos y hechos activados.

7.2.4 Justificación e Inspección de Soluciones

Permite explicar las decisiones con trazas y justificaciones:
- ¿Por qué?: objetivos globales.
- ¿Cómo?: cadena de razonamiento.

Puede ser texto predefinido o generado según contexto.

7.2.5 Aprendizaje
Los SBC pueden crear o modificar reglas al detectar errores (aprendizaje basado en explicaciones) o generar nuevas reglas tras observar cómo resolvieron problemas previos. También pueden aprender modelos por observación de ejemplos (aprendizaje inductivo).

7.3 Arquitectura de Sistemas Basados en Casos (CBR)
La solución se obtiene reutilizando casos anteriores similares. Reduce la necesidad de extraer y formalizar conocimiento.
Ciclo del CBR:
1. Recuperación: buscar casos similares.
2. Reuso: usar la solución del caso recuperado.
3. Revisión: adaptar y evaluar la solución.
4. Retención: decidir si se guarda el nuevo caso.

Ventajas del CBR:
- Menor formalización de conocimiento.
- Fácil mantenimiento (añadir/corregir casos).
- Más rápido al reducir razonamiento.
- Explicaciones más naturales (basadas en casos reales).

7.3.1 Almacenamiento del Conocimiento
Incluye:
- Base de casos: soluciones pasadas representativas.
- Conocimiento del dominio: para calcular similitud, adaptar y evaluar soluciones.

Casos organizados jerárquicamente, con indexación y cálculos de similitud (distancia).

7.3.2 Uso e Interpretación
Recupera casos similares, adapta soluciones y las verifica. Puede requerir razonamiento adicional si la solución no es válida directamente.

7.3.3 Almacenamiento del Estado
Guarda el caso actual, el caso recuperado y la información generada durante la recuperación y adaptación.

7.3.4 Justificación e Inspección
La justificación se basa en el caso utilizado y cualquier modificación hecha. Es intuitiva para el usuario.

7.3.5 Aprendizaje
Aprende añadiendo nuevos casos a la base. Se evalúa la utilidad de cada nuevo caso y se pueden eliminar casos irrelevantes.

7.4 Comunicación con el Usuario/Entorno
La interfaz debe permitir:
- Introducir datos.
- Preguntar al sistema su estado interno.
- Recibir preguntas del sistema.

SBC complejos pueden comunicarse con otros SBC usando lenguajes estandarizados y ontologías compartidas.

7.5.1 Redes Neuronales
Modelo conexionista, no simbólico. Formadas por neuronas artificiales organizadas en capas. Aprenden asociando entradas con salidas mediante ejemplos. Principal desventaja: no pueden justificar decisiones.

7.5.2 Razonamiento Basado en Modelos
Construye un modelo cualitativo del sistema (no numérico), ideal para problemas complejos. Ejemplo: física naïf, razonamiento de sentido común.

7.5.3 Agentes Inteligentes / Multiagente
Dividen tareas complejas entre agentes simples que se comunican y colaboran. Cada agente tiene percepción, razonamiento y acción. Requiere técnicas especiales: comunicación, negociación, planificación conjunta, cooperación.

Capítulo 8: Ingeniería de Sistemas Basados en el Conocimiento (SBC)

8.1 Ingeniería de SBC: Objetivo y desafíos
El objetivo central al desarrollar un SBC es transferir el conocimiento de un experto a un sistema computacional, incluyendo tanto los elementos del dominio como las metodologías de resolución.
- Intervienen dos roles:
  - Experto: Posee el conocimiento, no usa formalismos.
  - Ingeniero del conocimiento: Transforma ese conocimiento en una representación computable.

El desarrollo debe integrarse dentro de metodologías de ingeniería de software, adaptadas a las características específicas de los SBC.

8.1.1 Modelo en Cascada

Fases secuenciales:

1. Análisis del problema
2. Especificación de requerimientos
3. Diseño
4. Implementación
5. Prueba
6. Mantenimiento

Desventajas:

- No se visualiza el resultado hasta el final.
- No se puede usar ninguna parte del sistema hasta que se complete.

8.1.2 Modelo en Espiral

Integra cascada, prototipado y análisis de riesgos:

1. Identificación de objetivos y restricciones.
2. Evaluación de riesgos.
3. Formulación de estrategias para resolver riesgos.
4. Análisis del estado del desarrollo y planificación de siguientes pasos.

Clave:

Identificar riesgos tempranamente para evitar impactos negativos.

8.1.3 Diferencias clave entre SBC y software tradicional

1. Tipo de conocimiento:
    - SBC: conocimiento heurístico, incompleto, especializado.
    - Tradicional: algoritmos conocidos.
2. Adquisición del conocimiento (Knowledge Elicitation):
    - Interacción mediante entrevistas entre experto e ingeniero.
    - Desafíos: vocabulario técnico, formalismo adecuado, paradoja del experto (más experiencia, menos capacidad para explicar).
3. Estimación de conocimiento:
    - En SBC es difícil medir cantidad y naturaleza del conocimiento necesario.
    - Solución: prototipado rápido y desarrollo incremental.

8.1.4 Ciclo de vida de un SBC

Fases adaptadas (ver figura 8.5):

1. Análisis del problema
2. Especificación de requerimientos
3. Diseño preliminar
4. Prototipo inicial y evaluación
5. Diseño final
6. Implementación (desarrollo incremental)
7. Validación y verificación
8. Ajustes de diseño
9. Mantenimiento

8.1.5 Metodologías especializadas

CommonKADS:

- Ciclo en espiral.
- Utiliza 6 modelos interrelacionados: Organización, tareas, agentes, comunicación, conocimiento, diseño.
- Similar a UML.

MIKE:

- Ciclo en espiral: adquisición, diseño, implementación, evaluación.
- Usa lenguaje KARL para formalización y DesignKARL para diseño detallado.

8.2 Metodología sencilla para SBC pequeños/medianos

Fases:

1. Identificación del problema
2. Conceptualización
3. Formalización
4. Implementación
5. Prueba

8.2.1 Identificación

- Evaluar si se puede resolver con SBC (no algorítmico).
- Fuentes de conocimiento: expertos, libros, casos resueltos.
- Primer esquema del problema: Objetivos, motivación, estrategias, fuentes, tipos de tareas.

8.2.2 Conceptualización

- Visión del problema desde el experto.
- Determinar:
  - Conceptos y características → base para ontología.
  - Inferencias necesarias.
  - Subproblemas y jerarquía.
  - Flujo de razonamiento (qué se necesita y cuándo).
  - Distinguir evidencias, hipótesis y acciones.
- Se obtiene un modelo semiformal del dominio y de la resolución.

8.2.3 Formalización
- Visión desde el ingeniero.
- Elegir formalismo de representación (frames, reglas, lógica, etc.).
- Construcción de la ontología.
- Elegir técnicas de resolución (clasificación, razonamiento causal...).
- Tratar:
  - Incertidumbre, completitud, fiabilidad de información.
  - Identificar hechos seguros y hechos inciertos.

8.2.4 Implementación
- Implementar la ontología y las metodologías de resolución.
- Seleccionar herramientas adecuadas (ej. PROLOG, CLIPS...).
- Prototipo inicial para probar decisiones.
- Desarrollo incremental de módulos y funcionalidades.

8.2.5 Prueba
- Usar casos representativos (diseño y nuevos).
- Validar el funcionamiento y detectar fallos.
- Problemas comunes: Errores en diseño, implementación o formalización del conocimiento.
- Validación más difícil que en software tradicional por naturaleza heurística.

Capítulo 9: Resolución de Problemas en los SBC

9.1 Clasificación de los SBC
La construcción de un SBC es compleja, por lo que clasificar estos sistemas según las tareas que realizan permite:
1. Identificar tareas típicas.
2. Aplicar metodologías de resolución adecuadas.
3. Elegir métodos de representación e inferencia eficaces.

Primera clasificación funcional (1983):
- Interpretación: Inferir descripciones de situaciones a partir de observaciones.
- Predicción: Inferir consecuencias de situaciones/eventos.
- Diagnóstico: Detectar fallos desde síntomas.
- Diseño: Crear configuraciones que cumplan restricciones.
- Planificación: Generar acciones para alcanzar objetivos.
- Monitorización: Vigilar comportamiento frente a especificaciones.
- Corrección: Proponer soluciones a fallos.
- Control: Anticipar problemas y gobernar un sistema.

Problema: Superposición entre categorías.

Clasificación alternativa: operaciones genéricas
1. Análisis (Interpretar):
   - Identificación: Determinar tipo de sistema.
   - Predicción: Determinar resultados.
   - Control: Qué entradas logran salidas deseadas.
   - Monitorización: Detectar fallos.
   - Diagnóstico: Explicar fallos.
2. Síntesis (Construir):
   - Especificación: Qué restricciones debe cumplir.
   - Diseño: Generar configuraciones válidas.
   - Configuración: Estado actual del sistema.
   - Planificación: Pasos para ensamblar estructura.
   - Ensamblaje: Unir piezas del sistema.

9.2 Métodos de resolución de problemas
Dos técnicas generales principales:
1. Clasificación heurística
2. Resolución constructiva

9.2.1 Clasificación heurística
Asociar un problema a una solución predefinida, realizando inferencias intermedias. Es útil para tareas de análisis donde hay un espacio finito de soluciones.
Fases:

1. Abstracción de datos: Transformar datos del problema a atributos relevantes.
2. Asociación heurística: Buscar coincidencia entre problema y solución mediante experiencia.
3. Refinamiento: Ajustar solución abstracta a una solución concreta.

Tipos de abstracción:
- Definicional: Extraer características esenciales.
- Cualitativa: Convertir datos numéricos a categorías (ej. “alta”).
- Generalización: Usar jerarquías conceptuales.

Implementación en sistemas de reglas:
Proceso iterativo entre experto e ingeniero del conocimiento (IC):
1. Experto propone reglas.
2. IC las implementa.
3. Se prueban casos conocidos.
4. Si hay errores, se ajustan reglas.
5. Se prueban casos nuevos.

Componentes del conocimiento:
- Hipótesis: Posibles soluciones.
- Síntomas: Características que describen hipótesis.
- Causas: Datos que generan síntomas.

Se crean reglas de abstracción (causas → síntomas) y reglas heurísticas (síntomas → hipótesis), con niveles de confianza y posibilidad de introducir conceptos intermedios.

Ejemplo: Concesión de créditos
- Datos: Avales, bienes, morosidad, tipo de empresa, etc.
- Abstracción: Apoyo financiero, bienes, fiabilidad, compromiso, viabilidad.
- Soluciones: Denegar, aceptar, aceptar con rebaja/interés preferente.
- Reglas: Transforman datos a atributos, y atributos a decisiones.

9.2.2 Resolución Constructiva

Cuando la solución no puede enumerarse, debe construirse paso a paso. Ejemplos: diseño, planificación, síntesis.
Requiere:
- Conocimiento de restricciones (físicas, temporales, de entrada/salida).
- Evaluación precisa de decisiones intermedias (no solo heurística general).

Estrategias:
1. Proponer y aplicar:
Pasos:
1. Inicializar objetivo.
2. Proponer operadores aplicables.
3. Podar operadores inútiles.
4. Evaluar operadores con criterios expertos.
5. Seleccionar el mejor.
6. Aplicarlo.
7. Evaluar si se alcanzó el objetivo.

Puede usarse resolución secuencial (paso a paso) o descomposición jerárquica (subproblemas). Esta última mejora eficiencia.

2. Mínimo compromiso:
Partir de una solución completa no óptima e ir mejorando.
- Se modifican elementos de la solución de forma que restrinjan lo mínimo las opciones futuras.
- Si se viola una restricción, se retrocede con la mínima corrección posible.

Ejemplo: Planificación de trayectoria de un robot
- Objetivo: Llegar a una puerta sin colisiones.
- Operadores: Mover, girar.
- Restricciones: No chocar, minimizar tiempo.
- Evaluación: Basada en acercamiento al objetivo y libertad de obstáculos.
- Aplicación: Mediante estrategia proponer y aplicar.