Ingeniería del Conocimiento

Departamento de Computación

Curso 2002-2003

Alumna: Laura M. Castro Souto
Profesoras: Amparo Alonso Betanzos
Bertha Guijarro Berdiñas

Índice general

Capítulo 1

La Ingeniería del Conocimiento

En este primer tema estableceremos algunos conceptos básicos relacionados con la asignatura que nos ocupa.

1.1. El conocimiento y su contexto

¿Qué es el *conocimiento*? Es difícil de concretar, pero podemos sin embargo distinguir claramente que no es lo mismo que un *dato* ni tampoco es el mismo concepto que el de *información*.

Podemos decir que:

Dato Conjunto de señales, símbolos, signos, que llegan a nuestros sentidos, sin que tengan que tener significado propio.

Información Datos que se agrupan y adquieren un significado que no va implícito en ellos, sino que se aprende a manejar.

Conocimiento Conjunto de datos e información que además tiene **sentido del propósito** (sirve para algo) y **capacidad de generar nuevo conocimiento** e información (e incluso acciones).

	Característica	Ejemplo
Datos	Sin interpretar	`...-...`
Información	Añade significado	`S.O.S.`
Conocimiento	Propósito y capacidad de generación	Alerta, emergencia, comenzar un rescate

Cuadro 1.1: Diferencias entre dato, información y conocimiento

La definición de lo que es *conocimiento* se hace más difícil aún si consideramos que puede depender del *contexto*. Obviamente, un físico y un ajedrecista no tendrán la misma concepción de lo que es conocimiento referente a sus respectivas actividades.

En el caso de la Ingeniería del Conocimiento[1], esta situación se agrava, puesto que su aplicación se realiza en dominios muy concretos y diferentes (lo "normal" es distinto en cada caso, por ejemplo, para diversos pacientes en un hospital).

A veces se define el *conocimiento* como "información sobre la información", puesto que hay que tener cierta información sobre la información que se va a manejar para poder usarla adecuadamente.

La representación explícita del conocimiento es clave para distinguir entre sistemas software clásicos y sistemas software basados en conocimiento.

Como ya hemos estudiado con anterioridad (ver [2]), se pueden establecer categorías en el conocimiento barajado, en relación al origen y procedencia del mismo con respecto al experto de quien lo extraemos:

- **Conocimiento público**, que puede obtenerse directamente a partir de fuentes típicas (manuales, libros), comúnmente aceptado y universalmente reconocido.
- **Conocimiento semipúblico**, explícito pero no universalmente reconocido ni comúnmente aceptado, utilizado casi de forma exclusiva por los especialistas del área concreta.
- **Conocimiento privado**, no explícito, no universalmente reconocido ni comúnmente aceptado, de marcado carácter heurístico, endógeno de cada uno, fruto de la propia experiencia.

Un sistema de conocimiento pretende familiarizarse con el conocimiento público, implementar el semipúblico y extraer el privado.

1.2. La ingeniería del conocimiento

Denominamos *ingeniería del conocimiento* al conjunto de conocimientos y técnicas que permiten aplicar el saber científico a la utilización del conocimiento, entendiendo "conocimiento" como inteligencia o razón natural.

Partiendo de la siguiente definición de *ingeniería del software*[2] (IEEE-99):

> *La IS es la aplicación de una aproximación sistemática, disciplinada y cuantificable, al desarrollo, funcionamiento y mantenimiento del software (aplicaciones)*

podemos adaptarla a la IC y decir, asimismo, que la IC es la aplicación de una aproximación sistemática, disciplinada y cuantificable, al desarrollo, funcionamiento y mantenimiento del conocimiento (en aplicaciones, software).

[1] En adelante, IC.
[2] En adelante, IS.

Es por ello que muchas de las metodologías utilizadas en el campo de la IS se han adaptado a la IC, y que también muchos de los problemas que aparecen en una se reproducen en la otra también. La mayoría de ellos, como veremos en el tema correspondiente, se deben con frecuencia a la especificación débil de requisitos, a la presencia de entradas inconsistentes, etc. que dan lugar a diseños no limpios.

Resumiendo, podemos definir la *ingeniería del conocimiento* como el conjunto de principios, métodos, técnicas y herramientas que permiten la construcción de sistemas computacionales inteligentes.

1.3. Los sistemas basados en el conocimiento

Entendemos por *sistema basado en conocimiento*[3] un sistema computerizado (software) que utiliza y representa **explícitamente** conocimiento sobre un **dominio concreto** para realizar una tarea que requeriría un experto (de ser realizada por un humano), es decir, que es capaz de exportar ese conocimiento a través de los mecanismos apropiados de razonamiento para proporcionar un comportamiento de alto nivel en la resolución de problemas en ese dominio (Guida & Tasso).

Las dos características clave, tal y como se ha señalado, son:

- ▷ la representación explícita del conocimiento, algo que diferencia a los SBC de los sistemas software que se construyen en IS
- ▷ un dominio concreto, algo que los particulariza y diferencia de los sistemas de IA

A partir de la IA, surgieron una serie de ramas: la robótica, los sistemas conexionistas, los sistemas expertos,... Estos últimos fueron uno de los logros más importantes, porque fueron los primeros en enfrentarse a problemas reales utilizando conocimiento específico de pequeños dominios. No obstante, en los años 60 se produjo un retroceso en su desarrollo debido al aumento de la complejidad de los problemas que se pretendía abordar. Años más tarde, surgiría la IC.

Los beneficios más importantes que aportan los SBCs son:

- Mayor rapidez en la toma de decisiones
- Mayor calidad en la toma de decisiones
- Mayor productividad

El desarrollo de un SBC es caro para la empresa: se necesita contratar al menos un ingeniero de conocimiento, se va a consumir tiempo del experto... Si todo ello se compensa es por estas ventajas competitivas que acabamos de mencionar.

[3] A partir de ahora, SBC.

Hay varios conceptos que ayudan a distinguir los SBC de software más convencional y también de programas de inteligencia artificial:

- √ La naturaleza más bien heurística del conocimiento que contienen (IA, años 50-60).
- √ La naturaleza altamente específica del conocimiento (Dendral, 1970).
- √ La separación del conocimiento de cómo se usa —control— (General Problem Solver de McCarthy, 1963; Mycin, 1976).

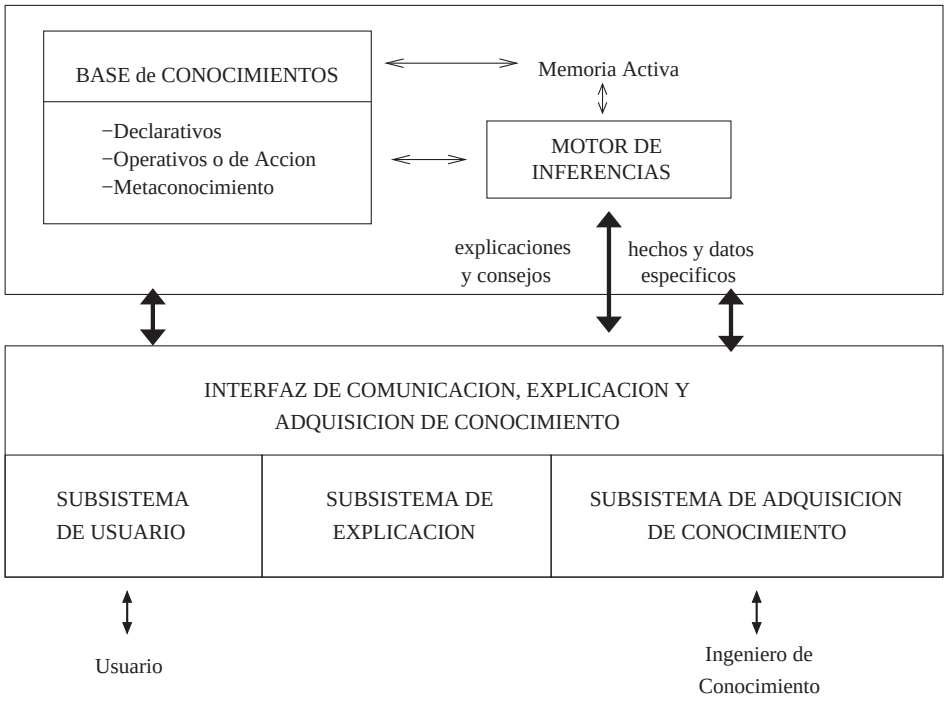


Figura 1.1: Esquema de un SBC.

Capítulo 2

Metodologías para la construcción de SSBBC

El ingeniero de conocimiento debe:

- √ elicitar
- √ estructurar
- √ formalizar
- √ operacionalizar

toda la información y el conocimiento que estén relacionados con el dominio. Ésta no es una tarea trivial, porque el conocimiento no es algo que se pueda observar, la información procede a menudo de diversas fuentes, se presenta en diferentes formatos, puede incluso ser a veces contradictoria. Es necesario organizar de alguna manera el trabajo a realizar.

Además, el conocimiento no es sólo algo difícil de extraer, sino también un recurso caro; por ello en los últimos tiempos ha surgido la idea de *reutilización* del conocimiento.

2.1. Diferencias entre la IS y la IC

Los ingenieros de conocimiento y los ingenieros de software estuvieron enfrentados durante mucho tiempo, hasta que se dieron cuenta de no había motivo para la confrontación, pues la IS no incluye a los sistemas de IC, ésta desarrolla su software en problemas mal estructurados o mal definidos que no son tratables en IS.

En IS el cliente pide lo que quiere; en IC se hacen modelos computacionales de un ámbito concreto, se hace un análisis exhaustivo de la organización donde vamos a aplicar nuestro modelo.

Las especificaciones de requisitos en IS son completas antes de empezar; en IC esto es casi imposible.

En IC es muy importante la adquisición del conocimiento, que además es continua, es el cuello de botella de todos los sistemas; en IS se adquiere todo lo que se necesita para funcionar.

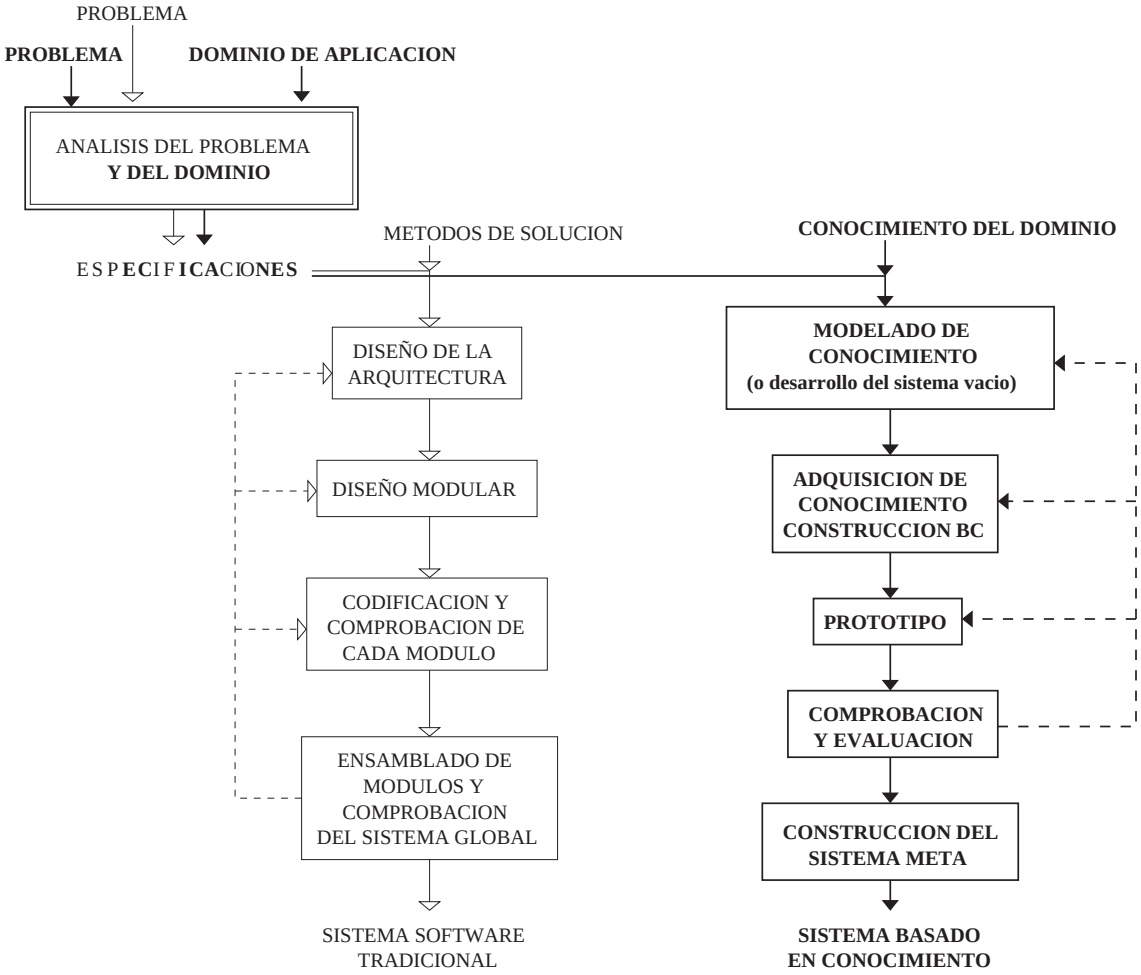


Figura 2.1: IS vs. IC.

En IC no se trabaja con lenguajes, sino con herramientas, ya que se ha conseguido que el control, el manejo del conocimiento sea genérico (i.e. motores de inferencias).

2.2. Metodologías adaptadas de la IS

En esta sección revisaremos superficialmente algunas de las metodologías que la IC ha "heredado" de la IS.

2.2.1. Metodología de prototipado rápido

Esta metodología consiste en adquirir conocimientos y codificar hasta considerar que tenemos un modelo lo suficientemente bueno.

Tras una serie de entrevistas con los clientes, usuarios y/o expertos, se intenta ver si el dominio puede:

- tener una parte "central" de la que puedan colgarse las demás posteriormente
- tener varias partes que se puedan tratar inicialmente por separado y comenzar con una de ellas

Si el contexto es favorable, se desarrolla un prototipo rápido para mostrar al experto, que se irá refinando y ampliando.

Las ventajas de esta metodología residen en que la rapidez en el desarrollo de una primera versión del sistema motiva al experto (pronto se ve algo operativo), y además ayuda a centrar el desarrollo del conocimiento además de no requerir demasiada experiencia. No obstante, desde el punto de vista de la IC son más importantes las desventajas que se presentan:

- dificulta la recogida de requisitos
- se sustituye el estudio de especificaciones y el diseño por la codificación rápida, lo que origina debilidades
- el crecimiento incontrolado complica la BC
- las interacciones no contempladas entre distintas partes o módulos del sistema son fuente de muchos problemas, el modelo crece distorsionado
- el código resulta generalmente muy poco y mal estructurado
- no se produce un análisis completo de requisitos
- no hay una buena documentación (o ésta es nula)
- el mantenimiento es prácticamente imposible

Esta metodología descuida todo lo que no tiene que ver directamente con el *core* del conocimiento (desarrollo de una IU, comunicación con otro software,...), con todo lo que ello conlleva.

2.2.2. Metodologías de desarrollo incremental

Ante el desbordamiento de la metodología de prototipado rápido, se volvió la vista a la IS y una de las primeras metodologías que se adoptó fue la de desarrollo incremental.

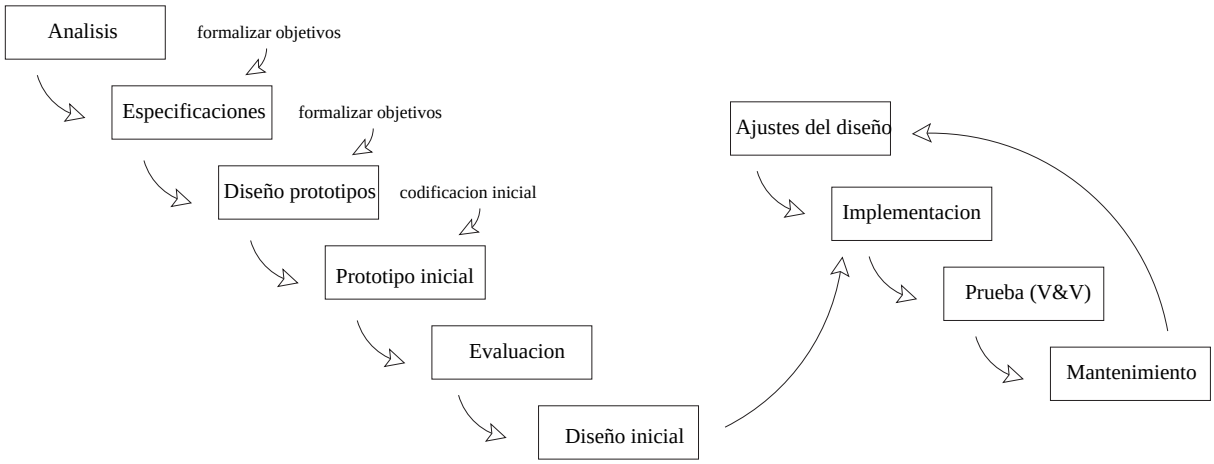


Figura 2.2: Esquema de la metodología de desarrollo incremental.

Aunque por incremental es más ordenada (manteniendo a la par algunas de las ventajas del prototipado rápido, como la pronta obtención de un sistema y una buena comunicación con los expertos), esta metodología gira, no obstante, en torno a la implementación y aunque logró organizar un poco más los sistemas, no centraba tampoco la atención en la captura de requisitos y especificaciones, que sería más adecuado para un sistema basado en conocimiento, a la par que no dejaba lugar para una etapa ulterior de mantenimiento preceptivo.

2.2.3. Metodologías en cascada

También adaptada de la IS, esta metodología trata de ajustar el alcance de la iteración de desarrollo, que resultaba problemático en el caso anterior (ver figura 2.3, página 11).

A pesar de conseguir reducir los errores al analizar más el modelo, el mantenimiento sigue siendo demasiado complejo para un sistema basado en conocimiento.

2.2.4. Metodologías en espiral

De los modelos planos se pasó al modelo en espiral, que aporta un interesante análisis de riesgos, además e plantear las iteraciones como capas en lugar de como bloques cerrados.

Si bien en IS no se utiliza demasiado porque resulta muy pesado, en IC iba a resolver múltiples cuestiones: los prototipos sucesivos se van refinando y ampliando, se pueden añadir especificaciones en cada vuelta hasta llegar a concretar finalmente el elemento

de test. Permite situar el mantenimiento en un nivel adecuado gracias al mencionado análisis de riesgos. Cada fase ayuda a *completar* la anterior, en lugar de sólo *sumar*, que era más el enfoque de metodologías anteriores, sin que se alteren los fundamentos anteriores.

Fue, pues, uno de los modelos que mejor funcionó, aunque no es demasiado bueno al desarrollar SBC más grandes. No obstante, aún quedaban dos cuestiones por solucionar:

* la adquisición del conocimiento seguía siendo el cuello de botella
* la capacidad de explicación no estaba realmente presente, ya que conocimiento y motivos iban juntos, indivisiblemente

Debido a esto, los propios SBC no tenían conciencia de sus límites. Se necesitaba una metodología que estructurase el conocimiento de una forma más adecuada, al fin y al cabo, la verdadera diferencia entre IS e IC es el tratamiento, el manejo del conocimiento. Todas las metodologías empleadas hasta el momento lo encuadraban en un lugar o en otro, tratándolo sin darle un nivel específico como es imperativo. Como consecuencia de estos problemas, los SBC eran por añadidura muy difíciles de mantener, con fases de validación muy extensas.

Fue primero Newell el que dio la voz de alarma indicando la necesidad de tratar el conocimiento como algo especial, reflexionando sobre lo que hay que representar y cómo se quiere hacerlo, y posteriormente McDermott con su teoría sobre las "Tareas genéricas" según la que la adquisición del conocimiento sigue siempre unos pasos repetitivos, los que impulsarían el desarrollo de metodologías propias de la IC (con raíces, claro está, en las que acabamos de ver). De entre ellas, estudiaremos la **metodología CommonKADS**.

Newell. El nivel de conocimiento

El mayor problema detectado hasta el momento es la no-diferenciación de lo que es conocimiento de la representación del mismo. La solución pasa por añadir el *Nivel de Conocimiento*, que está por encima del nivel simbólico. En este nivel el sistema se comporta como un agente que tiene tres vistas:

- componentes ≡ objetivos
- acciones
- cuerpo (conocimientos sobre objetos y acciones)

El medio sobre el que actúa es el conocimiento: el agente toma el conocimiento, lo procesa y realiza acciones para conseguir objetivos. Esto permitió también abordar las primeras ideas sobre reutilización del conocimiento: abstraer las tareas genéricas para volver a utilizarlas en sistemas similares.

Una metodología que usa el nivel de conocimiento es KLIC (*KBS Life Cycle*).

McDermott. Métodos de limitación de roles

Los estudios de McDermott constituyeron los primeros intentos para reutilizar el método de resolución de problemas.

Todos los sistemas tenían un motor de inferencias separado del conocimiento hasta ese momento. McDermott pensó que el problema de reutilizar el motor era que parte del conocimiento de control debía ir codificado en la base de conocimiento. Así, a la vez que se metía conocimiento declarativo nuevo se iba deteriorando el anterior.

Para evitarlo, McDermott propuso estudiar los métodos de resolución de problemas, diferenciándolos de la base de conocimientos. Como conclusión, extrajo que hay familias de tareas que se pueden resolver por métodos cuyo conocimiento de control es abstracto y se puede aplicar a distintas instanciaciones de esa tarea. Además, permite definir en qué orden hay que adquirir el conocimiento y también cómo se implementa (al ordenarlo, disminuímos la entropía y es más fácil implementarlo).

2.3. Metodología CommonKADS

La metodología **CommonKADS** (*Knowledge Analysis and Documentation System*) es una variación de la metodología en espiral de la IC, desarrollada en Europa en torno a 1983. Surge de su predecesora, KADS, al serle añadido un lenguaje de modelado conceptual (CML, *Conceptual Modell Language*), muy parecido a UML, y que facilita el diseño del sistema.

La metodología CommonKADS es, pues, una metodología estructurada que cubre la gestión del proyecto, el análisis de la organización, la ingeniería del conocimiento y del software. Plasma tres de las ideas más utilizadas en IS e IC (modelado, reutilización y gestión del riesgo) y, siendo la única que utiliza Orientación a Objetos, se organiza tal y como se puede observar en la figura 2.4 (página 11).

Esta división en niveles y modelos permite su desarrollo sin que unos sean interdependientes de otros, es decir, permite un gran nivel de desacoplamiento. El conocimiento se encuentra así perfectamente estructurado y documentado, pues cada modelo posee una serie de *plantillas* asociadas.

2.3.1. Nivel de Contexto: ¿Por qué?

Los modelos de este nivel analizan los beneficios, el impacto, la utilidad que tendrá el SBC que se pretende construir, su viabilidad, etc.

Modelo de Organización Estudia qué áreas de la organización son más susceptibles de sacar provecho de un SBC. Es un estudio profundo de la organización, del impacto y posibles resultados de la implantación del SBC, espectativas, predisposición,...

Modelo de Tareas Ubicadas las tareas más importantes, es el momento de intentar descomponer el/los sistemas en "primitivas" con el fin de

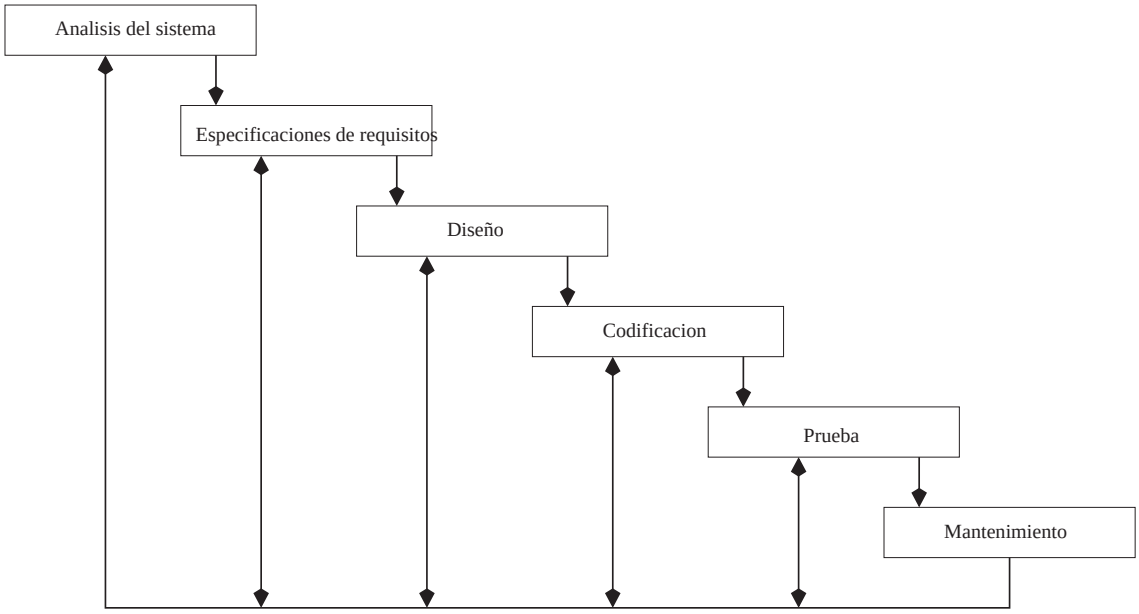


Figura 2.3: Esquema de la metodología en cascada.

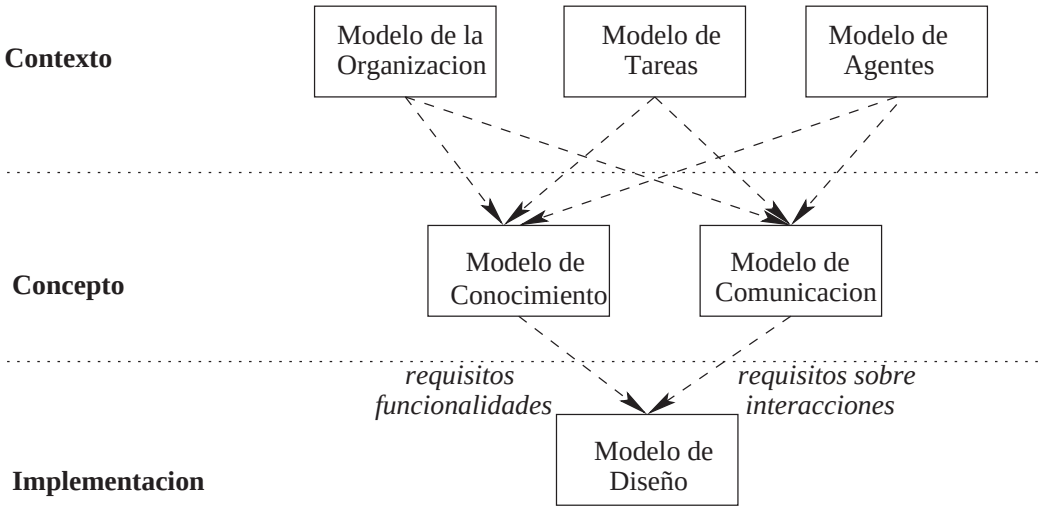


Figura 2.4: Niveles de la metodología CommonKADS.

poder abordarlas más fácilmente, identificar sus entradas y salidas, criterios de rendimiento, pre y postcondiciones,...

Modelo de Agentes Los agentes son los ejecutores de las tareas de la organización (ya sean personas físicas, sistemas de información, etc.). Se analiza en este modelo qué normas se les aplican, plantillas, funciones,...

2.3.2. Nivel de Concepto: ¿Qué?

Los modelos de este nivel conforman una descripción conceptual del conocimiento.

Modelo de Conocimiento Explica en detalle qué tipos de conocimiento e información tenemos involucrados (naturaleza y estructura). Da una visión de la estructura del conocimiento independiente de la implementación.

Modelo de Comunicación Disecciona cómo es la comunicación entre agentes involucrados (conceptualmente).

2.3.3. Nivel de Implementación: ¿Cómo?

Este nivel se centra en la manera de llevar a cabo, de realizar de manera concreta, el sistema: mecanismos computacionales, arquitectura, representación del conocimiento más adecuada, etc.

Modelo de Diseño Usando fundamentalmente el Modelo de Conocimiento y el Modelo de Comunicación, se intentan obtener los requisitos y restricciones prácticas del sistema.

Capítulo 3

Análisis de viabilidad e impacto: modelado del contexto en CommonKADS

El conocimiento siempre funciona dentro de una organización. En este capítulo, entre otras cosas, veremos:

- √ Por qué es necesario modelar el contexto
- √ El papel de los modelos: Organización, Tareas y Agentes
- √ Pasos y técnicas en el análisis del conocimiento orientado a las empresas y las instituciones
- √ Casos de ejemplo

Razones del modelado del contexto

- ⋆ A menudo es difícil *identificar el uso rentable* de tecnologías basadas en conocimiento.
- ⋆ El laboratorio es diferente del "mundo real".
- ⋆ La *aceptabilidad de los usuarios* es muy importante.
- ⋆ Un sistema sólo puede funcionar de forma adecuada si está propiamente *integrado* a largo plazo en la organización en la que trabaja.
- ⋆ Un SBC actúa como un *agente que coopera* con otros, humanos o no, y lleva a cabo una fracción de las tareas de la organización.
- ⋆ Un SBC es una *herramienta de apoyo* dentro del proceso general de la organización, al igual que cualquier sistema de información, en general.

Muchas de estas cuestiones no se tenían en cuenta en metodologías anteriores, lo que supone un gran avance.

Las metas del modelado del contexto son:

- Identificar qué cuestiones suponen problemas y cuáles no.
- Decidir soluciones y su viabilidad.
- Mejorar las tareas y el conocimiento relativo a éstas.
- Planificar la necesidad de cambios en la organización.

El papel de los SBC se rige por una serie de directrices:

- Normalmente los SBCs encajan mejor en proyectos de mejora de la organización, más que en la visión tradicional de automatizar las tareas del experto.
- Las tareas son normalmente demasiado complejas y el proyecto se suele convertir en un fracaso, a raíz de expectativas poco realistas.
- Es mejor usar los SBCs como herramientas de mejora de procesos.
- El papel típico de los SBCs es el de un asistente interactivo inteligente, a diferencia de la mayoría de los sistemas automáticos, que son pasivos.

3.1. El Proceso de Modelado del Contexto

Los pasos a seguir son:

1. Llevar a cabo un estudio de alcance y viabilidad. Herramienta: *Modelo de Organización* (OM).
2. Llevar a cabo un estudio de impacto y mejora (para enfocar/ampliar/refinar el modelo de la organización). Herramienta: *Modelos de Tareas y de Agentes* (TM, AM).

Cada estudio consta de una parte de análisis y una parte de decisión constructiva:

1. Estudio del alcance y viabilidad:
   a) Análisis.- Se trata de identificar las áreas problema/oportunidades y buscar soluciones potenciales, ubicándolos en una perspectiva más amplia en la organización.
   b) Síntesis.- Se trata de estudiar la viabilidad económica, técnica y del proyecto, elegir el área (o áreas) más comprometedora y la solución meta.
2. Estudio de impacto y mejoras (para cada área elegida en el paso anterior):
   a) Análisis.- Se estudian las interrelaciones entre la tarea, los agentes involucrados y el uso de conocimiento para un sistema con éxito, intentando ver qué mejoras se pueden lograr.

b) Diseño.- Se decide acerca de los cambios en las tareas y las medidas de la organización para asegurar su aceptación y la integración de una solución basada en SBC.

Como ya hemos visto en el capítulo anterior, el nivel contextual aglutina tres *modelos*:

- Estudio de alcance y viabilidad
  - *Modelo de la Organización* (OM) para describir y analizar la organización en sentido amplio
- Estudio de impacto y mejoras
  - *Modelo de Tareas* (TM) y *Modelo de Agentes* (AM), más centrados y detallados, enfocan las partes relevantes
  - TM: tareas y conocimiento relativo a ellas directamente relacionado con el problema a resolver con el SBC
  - AM: agentes involucrados en las tareas del TM

Para simplificar este trabajo se dispone de *formularios* u *hojas de trabajo* que ayudan en el proceso de modelado:

- ▷ 5 formularios para el OM
- ▷ 2 formularios para el TM
- ▷ 1 formulario para el AM
- ▷ 1 formulario resumen

Estas hojas de trabajo funcionan como "checklist" y como archivo de información, debiendo ser utilizados de forma flexible.

3.1.1. El modelo de Organización

Veremos ahora cómo analizar una organización intensiva en conocimiento:

- ∗ Describir aspectos de la organización
  - — carpeta de oportunidades/problemas
  - — contexto de negocio, metas, estrategia
  - — organización interna
    - √ estructura
    - √ procesos
    - √ personas (plantilla, roles funcionales,...)
    - √ potencial y cultura
    - √ recursos (conocimiento, sistema de soporte, equipos,...)
- ∗ Hacer este trabajo para la organización presente y la futura
- ∗ Comparar y tomar las primeras decisiones de qué hacer

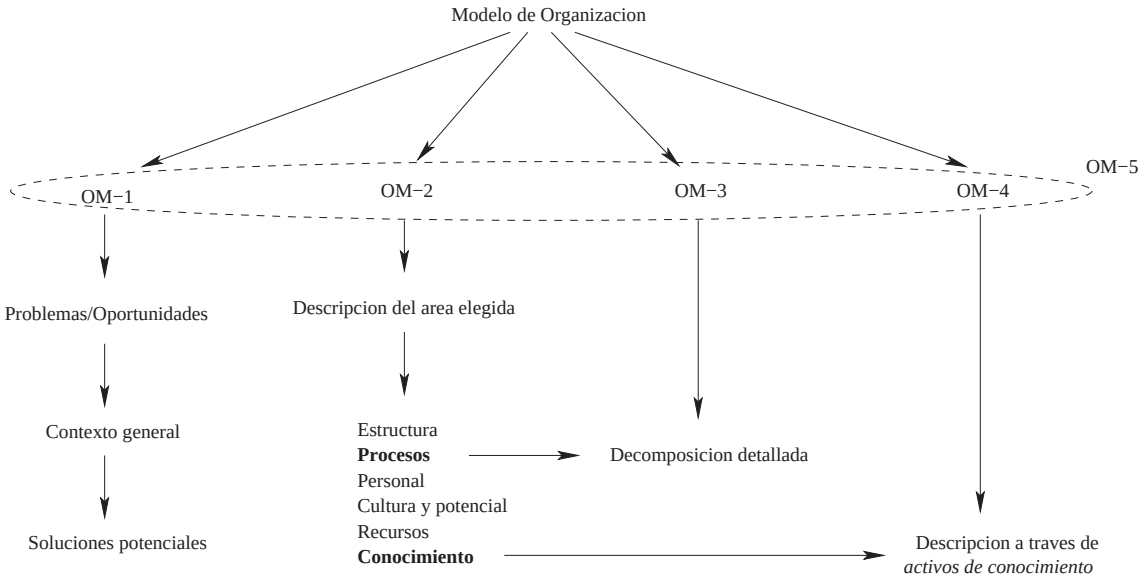


Figura 3.1: Modelo de la Organización.

Se remite a los apéndices para detalle de las plantillas correspondientes a cada paso del análisis del Modelo de Organización.

Hasta aquí, es el análisis de los aspectos estáticos de la organización, los que no se supone que vayan a cambiar.

3.1.2. El modelo de las Tareas

El *modelo de Tareas* describe, utilizando también una serie de plantillas que se adjuntan en los apéndices, las tareas que se determinan componen los procesos de la organización y que fueron esbozadas en algunos apartados referentes al modelo de la Organización.

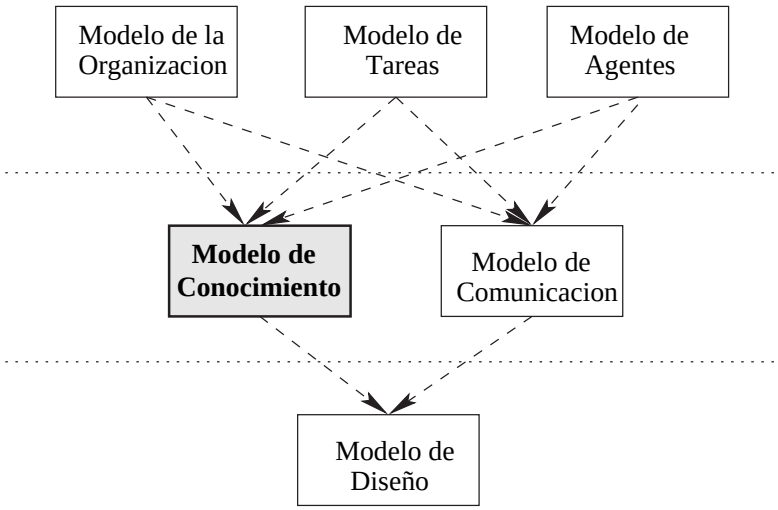
3.1.3. El modelo de los Agentes

Por su parte, el *modelo de Agentes* detalla el papel, relevancia, conocimiento y otras características relativas a los agentes que llevan a cabo o participan en las tareas identificadas en el modelo de Tareas. De nuevo, remitimos a los apéndices para estudio de las plantillas asociadas.

Capítulo 4

Descripción conceptual del conocimiento en CommonKADS

Como hemos visto, CommonKADS organiza la aproximación a un SBC de la siguiente forma:



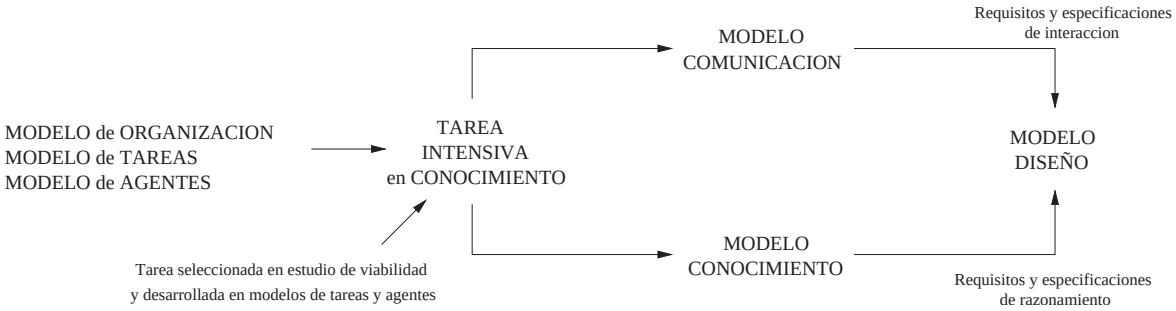
En este capítulo nos centraremos en el **modelado del Conocimiento**.

4.1. El modelo del Conocimiento

Los *modelos de Conocimiento* son una herramienta especializada para especificar tareas en dominios intensivos de/en conocimiento.

Un modelo de conocimiento especifica los requisitos de conocimiento y razonamiento del sistema futuro. No incluye aspectos de comunicación con los usuarios ni con otros agentes software, ni tampoco contiene términos específicos de implementación.

Su estructura es similar a la de los modelos de análisis tradicional en ingeniería del software, siendo un aspecto importante para la reutilización del software.



El término *conocimiento* ya ha sido comentado con anterioridad: lo habíamos definido como "información sobre la información". Un ejemplo de ello podría ser, por ejemplo, en las jerarquías superclase-subclase de tipos de objetos, un link entre dos clases, que proporciona información sobre la relación entre ambas.

El conocimiento se puede utilizar para inferir nueva información, de suerte que no hay realmente una frontera definida entre información y conocimiento.

En un SBC, el conocimiento está presente como tal en la *base de conocimientos*. Normalmente, se prefiere tener varias bases de conocimiento, cada una aglutinando reglas de un tipo determinado, de manera que sea posible su reutilización y también su corrección de forma más sencilla.

Dentro del modelo del conocimiento, distinguiremos varias *categorías de conocimiento*:

Conocimiento de la Tarea *¿Qué y cómo?*

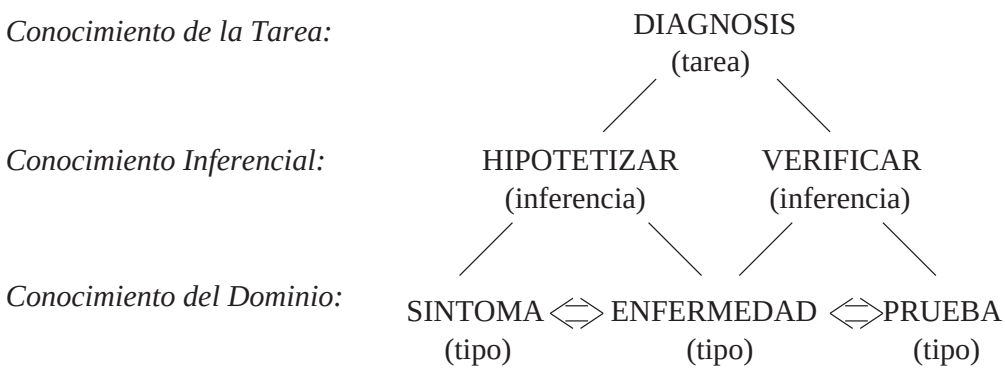
Es un conocimiento orientado a la meta y que realiza una descomposición funcional.

Conocimiento Inferencial

Encarna los pasos básicos del razonamiento que se pueden hacer en el dominio (en el contexto de un problema) y que se aplican mediante las tareas.

Conocimiento del Dominio

Aglutina el conocimiento y la información relevantes del dominio estático, equivaliendo de algún modo al modelo de datos o de objetos en IS.



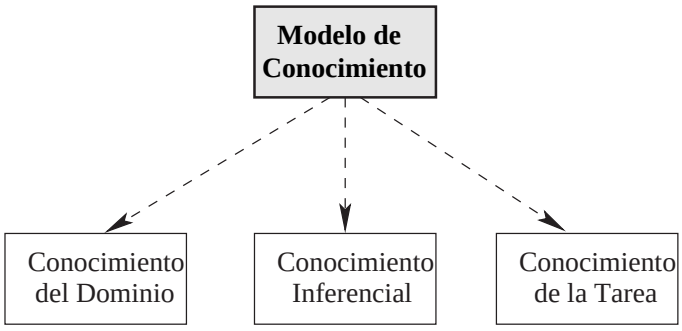


Figura 4.1: Categorías en el modelo del Conocimiento.

4.1.1. Conocimiento del dominio

El *conocimiento del dominio* describe la información estática más importante y los objetos de conocimiento en un determinado dominio.

Tiene dos partes principales:

Esquema del Dominio

Describe la estructura estática de la información/conocimiento a través de definiciones tipo, siendo comparable al modelo de datos/objetos en IS. Queda definido a través de los constructos del dominio.

Base de Conocimientos

Contiene instancias de los tipos que se especifica en el esquema del dominio (es decir, conjuntos de instancias de conocimiento), siendo comparable al contenido de una base de datos.

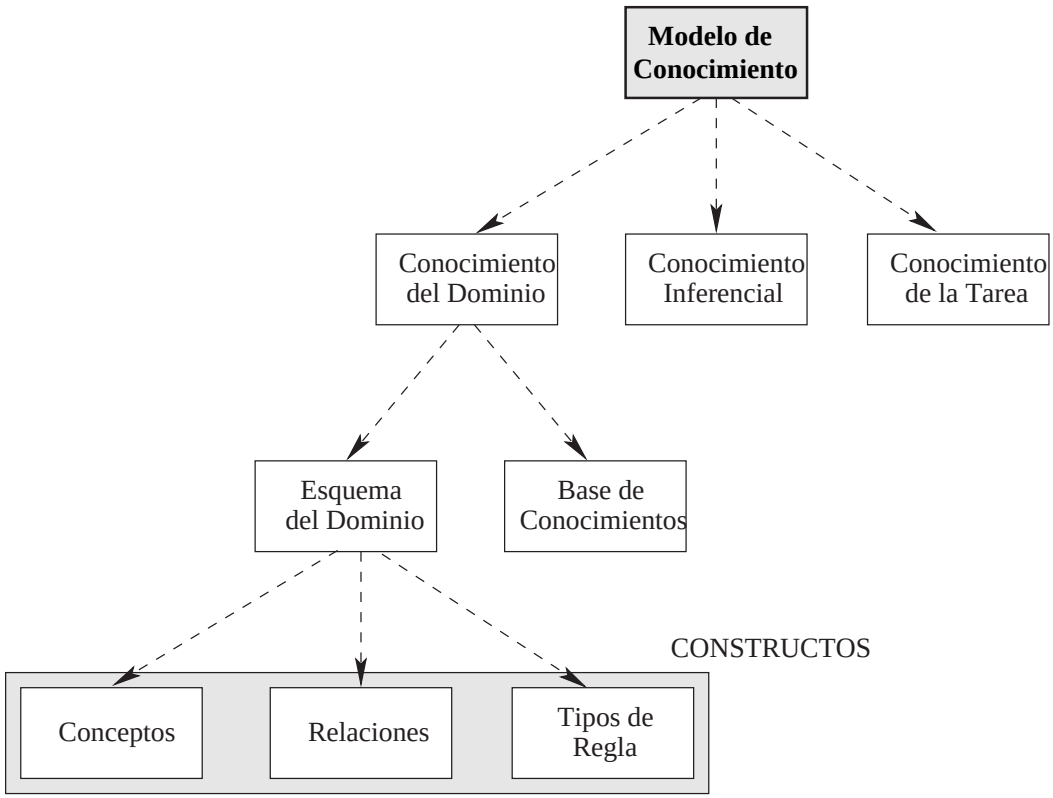


Figura 4.2: Constructos del modelo del Conocimiento.

Constructos en el esquema del dominio

La mayoría son similares a los de O.O. (especialmente los diagramas de clases):

Conceptos Describen un conjunto de objetos o instancias del dominio que comparten características similares (como los objetos en O.O. pero sin operaciones ni métodos).

Relaciones Como las asociaciones en O.O.

Atributos Valores primitivos. Características de los conceptos.

Tipos de reglas Introducen expresiones (no hay equivalente en IS).

Se incluyen, además, otros para cubrir aspectos específicos del modelado SBC.

Conceptos y Atributos

Como hemos dicho, un *concepto* describe un conjunto de objetos o instancias. Las características de los constructos se definen mediante *atributos*, que pueden tener un valor (atómico) que se define a través de un tipo de valor (definición de los valores permitidos).

Los conceptos son el punto de comienzo para el modelado del conocimiento.

Relaciones

Las *relaciones* entre conceptos pueden definirse con el constructo RELACIÓN o RELACIÓN BINARIA (e incluso RELACIÓN N-ARIA) a través de las especificaciones de argumentos.

La cardinalidad se define para cada argumento y su valor por defecto es 1. Es posible especificar un *rol* para cada argumento.

La propia relación puede tener atributos.

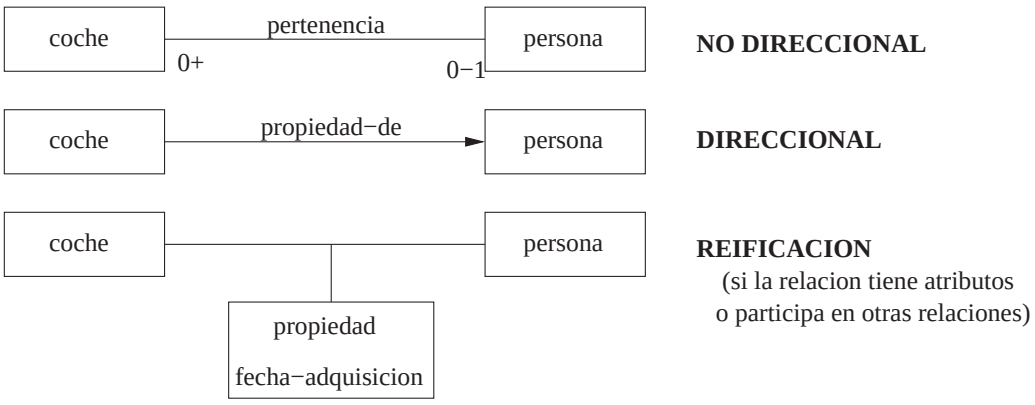


Figura 4.3: Relaciones en el modelo del Conocimiento.

El modelado de las reglas

Las reglas son una forma natural y común de representar el conocimiento simbólico. Ahora bien, ¿cómo representamos dependencias entre conceptos en un modelo de datos tradicional?

Para modelar la construcción de las reglas se usa el constructo TIPO DE REGLA:

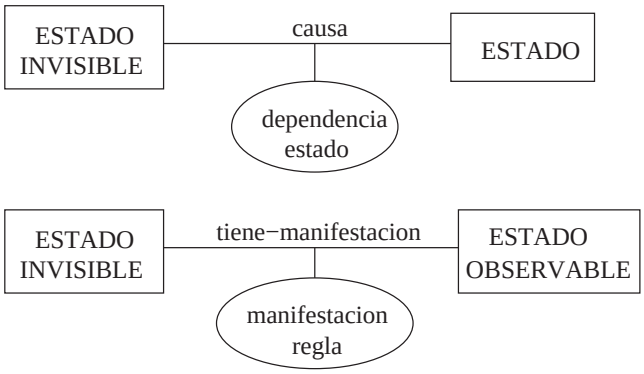
- Es similar a una relación, donde antecedente y consecuente no son instancias de conceptos sino expresiones de esas instancias.
- Se modela una relación entre expresiones acerca de los valores de los atributos.
- Se modelan un conjunto de reglas de estructura similar.
- Las relaciones no son estrictamente lógicas, es necesario especificar un SÍMBOLO DE CONEXIÓN entre antecedente y consecuente.

Estructura de tipo de regla

La estructura es sencilla:

```
<antecedente> <símbolo-de-conexión> <consecuente>
```

Su uso flexible permite la representación de cualquier tipo de dependencia (tipos múltiples para antecedente y consecuente).




```
TIPO-de-REGLA regla-manifestacion
  DESCRIPCION "..."
  ANTECEDENTE estado-invisible
  CONSECUENTE estado-observable
  SIMBOLO     tiene-manifestacion
END-TIPO-de-REGLA
```

Figura 4.4: Ejemplos de representación de Tipos de Regla.

Base de Conocimientos

Es una partición conceptual de la BC que contiene instancias de los tipos de conocimiento definidos en el esquema del dominio.

Las instancias de los tipos de reglas contienen reglas. Su estructura tiene dos partes:

- el SLOT USA: `<tipos-usados>de <esquema>`
- el SLOT EXPRESIONES: `<instancias>`

Las instancias pueden representarse formalmente, o bien semiformalmente con el símbolo de conexión separando antecedente y consecuente.

```
BASE-CONOCIMIENTOS
  USA ... de ...
      ... de ...
  EXPRESIONES
      /* dependientes-estado */
      ...
      /* manifestacion-regla */
      ...
END-BASE-CONOCIMIENTOS
```

Figura 4.5: Ejemplo de representación de Base de Conocimientos.

4.1.2. Conocimiento inferencial

El *conocimiento inferencial* describe el nivel inferior de descomposición funcional. Describe cómo las estructuras estáticas del conocimiento del dominio se pueden usar para realizar el proceso de razonamiento, permitiendo la reutilización del conocimiento.

Sus elementos principales se aprecian en la figura 4.6 y son:

Inferencias Relacionadas con el razonamiento, son las unidades básicas de procesado de información.

Funciones de Transferencia Relativas a la comunicación con otros agentes (a un nivel muy básico, esta cuestión se trata realmente en el Modelo de Comunicación).

Roles de Conocimiento Relacionados indirectamente con el conocimiento del dominio.

Una inferencia usa el conocimiento de alguna base de conocimiento para derivar nueva información.

Los roles dinámicos son las entradas y salidas en tiempo de ejecución de las inferencias.

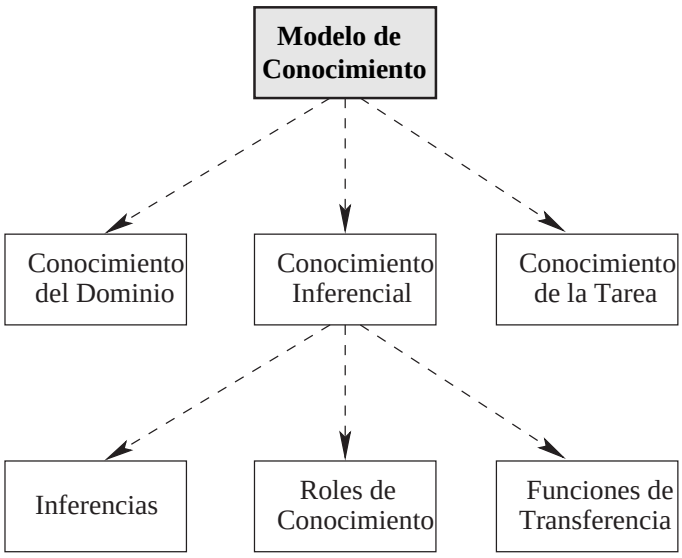


Figura 4.6: Elementos del Conocimiento Inferencial.

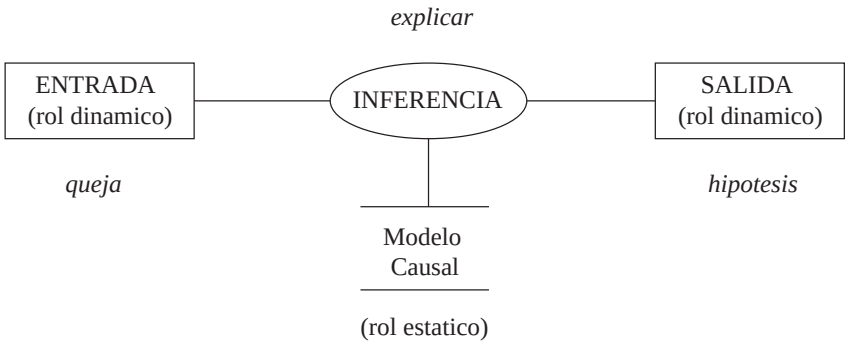


Figura 4.7: Ejemplo de Inferencia.

Inferencias

Las *inferencias* quedan completamente descritas a través de una especificación declarativa de propiedades de su E/S. Los procesos internos de la inferencia son una caja negra.

Rol de Conocimiento

Proporcionan un nombre funcional para elementos dato/conocimiento. Dicho nombre captura el rol del elemento en el proceso de razonamiento, realizando un mapeado explícito a los tipos del dominio.

Los roles dinámicos son variantes E/S, mientras que los estáticos son entradas invariantes (una base de datos).

```
INFERENCIA explicar
  ROLES
   ENTRADA   ...
   SALIDA    ...
   ESTATICOS ...
  ESPECIFICACION "..."
END INFERENCIA
```

```
ROL-CONOCIMIENTO nombre
   TIPO dinamico
   MAPEADO-DOMINIO visible-estado
END-ROL-CONOCIMIENTO
```

Funciones de Transferencia

Las *funciones de transferencia* transfieren un item de información entre el agente de razonamiento del módulo de conocimiento y otro agente del mundo externo (usuario, otro sistema,...).

Desde el punto de vista del modelo de conocimiento es una caja negra: sólo interesa su nombre y su E/S. La especificación detallada de las funciones de transferencia es parte de otro modelo, el de Comunicación.

	Iniciativa sistema	Iniciativa externa
Información externa	OBTENER	RECIBIR
Información interna	PRESENTAR	PROPORCIONAR

Cuadro 4.1: Nombres estándar de las Funciones de Transferencia.

Estructura Inferencial

La combinación de los diferentes conjuntos de inferencias especifica la capacidad inferencial básica del sistema en desarrollo. El conjunto de inferencias se puede presentar gráficamente en una ESTRUCTURA INFERENCIAL, que hace énfasis en los aspectos dinámicos del flujo de datos (roles estáticos opcionales).

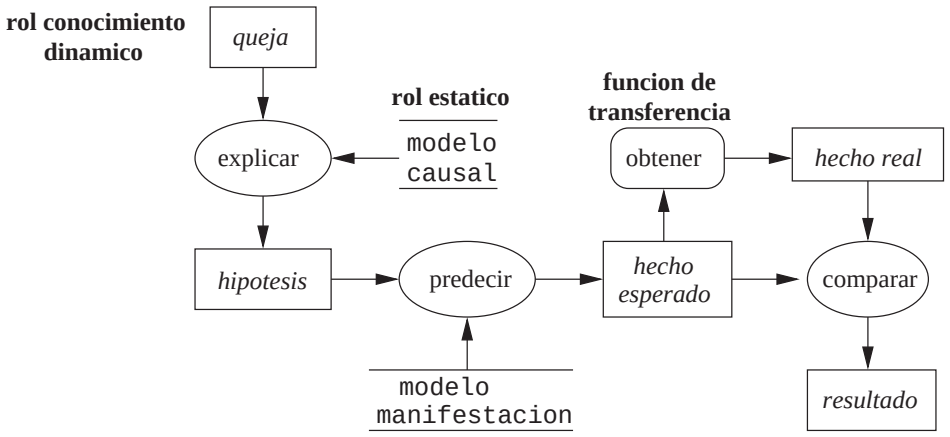


Figura 4.8: Ejemplo de Mapa Inferencial.

Uso de las Estructuras Inferenciales

Las *estructuras inferenciales* son representaciones abstractas de los posibles pasos del proceso de razonamiento, y, como tales, son un vehículo importante de comunicación durante el proceso de desarrollo, a pesar de que a menudo puedan ser provisionales.

Suele ser útil realizar anotaciones con ejemplos específicos del dominio.

Las estructuras inferenciales definen las capacidades inferenciales del sistema, el vocabulario y las dependencias de control, pero no el control en sí (del que se ocupa el conocimiento).

Reutilización de inferencias

El estado ideal sería disponer de un conjunto estándar de inferencias. Con ese objetivo, se recomienda el uso de nombres lo más estándar posibles con el fin de favorecer la reutilización.

4.1.3. Conocimiento de la tarea

El *conocimiento de la tarea* describe metas (por ejemplo, asesorar la suscripción de una hipoteca, diseñar un ascensor,...) y las estrategias que se pueden utilizar para realizar dichas metas. Esta descripción sigue un esquema jerárquico.

Tal y como se puede observar en la figura 4.9, distinguiremos, dentro del conocimiento de la tarea, la propia *Tarea* y por otra parte lo que llamaremos el *Método de la Tarea*.

Tarea

La *Tarea* define la meta del razonamiento en forma de pares (entrada, salida), con el fin de especificar qué es necesario saber.

La diferencia principal con las funciones tradicionales es que los datos manipulados por la tarea se describen también independientemente del dominio.

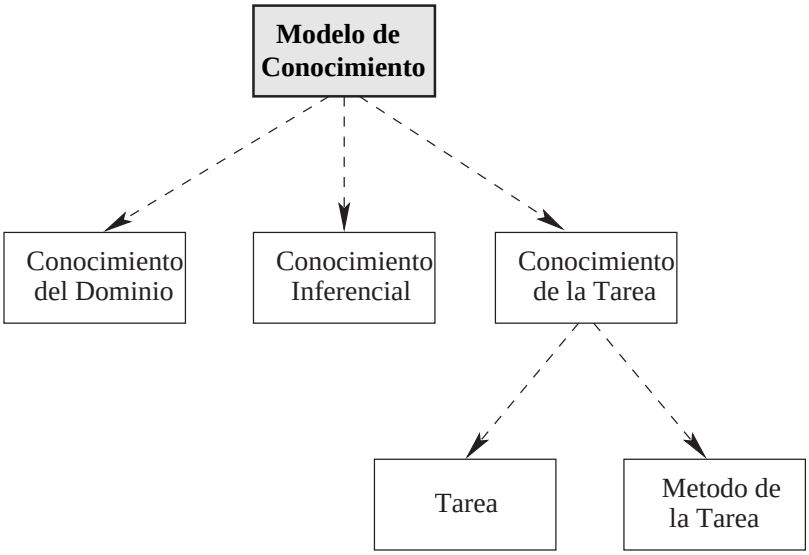


Figura 4.9: Elementos del Conocimiento de la Tarea.

El hecho de que la descripción deba ser independiente del dominio tiene como objetivo la reutilización de las tareas.

Una tarea se describe por medio de tres slots:

META Descripción textual informal.

SPEC Describe de manera textual e informal la relación entre la entrada y la salida de la tarea.

ROLES

Los roles de E/S se especifican en forma de roles funcionales, como en las inferencias, pero con algunas diferencias:

- no hay roles estáticos
- no hay mapeado de los roles en términos específicos del dominio; los roles de las tares están linkados a los roles inferenciales
- cada tarea tiene un método asociado

Método de la Tarea

El *Método de la Tarea* describe cómo se realiza una tarea mediante su descomposición en subfunciones. Las subfunciones pueden ser otra tarea, inferencias o funciones de transferencia.

La parte central del método de la tarea se denomina *estructura de control* y describe el orden de las subfunciones, capturando la estrategia de razonamiento.

Los elementos de la estructura de control son:

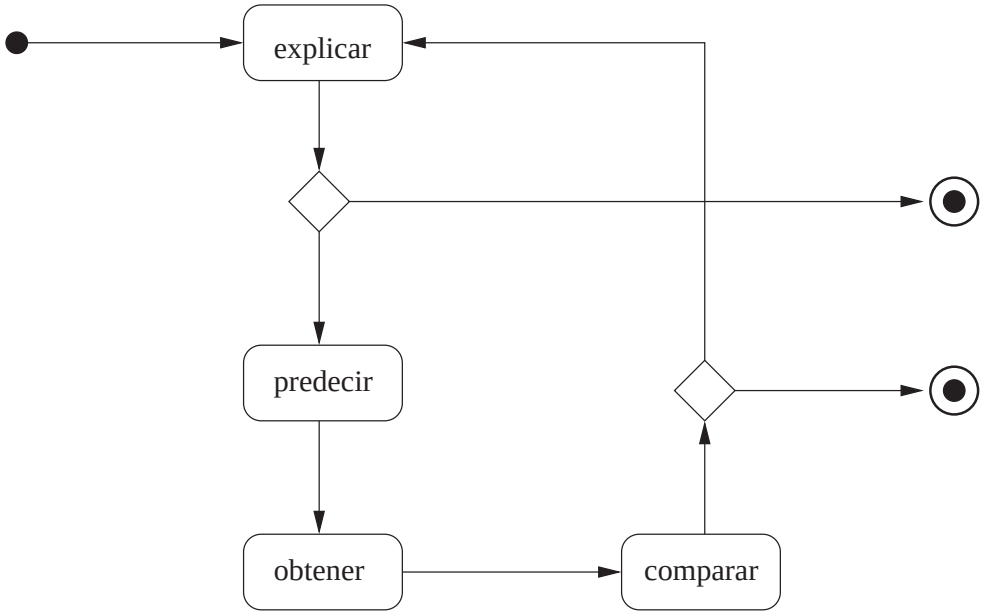


Figura 4.10: Ejemplo de esquema de un posible Método de la Tarea.

- llamadas a procedimientos (tareas, inferencias, funciones de transferencia)
- operaciones de roles (asignación, suma/resta,...)
- primitivas de control (repetir,...)
- condiciones
  - expresiones lógicas sobre roles
  - condiciones especiales: `tiene_solución` y `nueva_solución`

4.1.4. ¿Inferencia o Tarea?

Si el comportamiento interno de una función[1] es importante para explicar el comportamiento del sistema como un todo, entonces es necesario definir esta función como una tarea.

Durante el desarrollo del modelo, es usual manejar estructuras inferenciales provisionales.

4.1.5. Modelo de Datos (IS) vs. Modelo de Conocimiento (IC)

Los *Modelos de Datos* contienen "datos sobre datos", ya que en Ingeniería del Software lo importante son los datos. Sin embargo, la Ingeniería del Conocimiento se centra en el conocimiento, hace énfasis en el control interno y desarrolla funciones que se describen independientemente del modelo de datos, lo que favorece una mayor reutilización posterior.

[1]Donde por *función* podemos entender tanto "tarea" como "inferencia".

4.2. Plantillas de modelos de Conocimiento. Elementos reutilizables

En lo que llevamos visto hasta el momento, destacan una serie de puntos clave en lo que a reutilización se refiere:

- √ Los modelos de conocimiento pueden ser parcialmente reutilizados en aplicaciones nuevas.
- √ La guía principal para la reutilización es el *tipo de tarea*.
- √ Existe un **catálogo de plantillas de tareas** intensivas en conocimiento (como los patrones en O.O.).

Los fundamentos de la reusabilidad pasan por no reinventar la rueda cada vez que nos enfrentamos a un problema, conseguir la máxima eficiencia coste/tiempo, disminuir la complejidad y asegurar la calidad.

Una **plantilla** es una combinación de elementos del modelo reutilizables:

- Estructura inferencial (provisional)
- Estructura de control típica
- Esquema del dominio típico desde el punto de vista de la tarea

Todo ello es específico para el tipo de tarea que describe cada plantilla en particular. Gracias a estas plantillas este método de modelado soporta el modelado del conocimiento "top-down".

4.2.1. Tipos de Tareas

El rango de tipos de tareas está limitado. Esto es una ventaja de la IC en comparación con los antiguos SSEE.
En el trasfondo de esto se encuentran la ciencia cognitiva y la psicología.

La estructura de la descripción en la plantilla es la siguiente:

1. Caracterización general.
2. Método por defecto.
3. Variaciones típicas (cambios/ajustes frecuentes).
4. Esquema típico del conocimiento del dominio (asunciones sobre las estructuras del dominio).

	Tareas Analíticas	Tareas Sintéticas
Sistema[a]	Preexiste (aunque no es conocido).	No existe aún.
Entrada	Datos acerca del sistema.	Requisitos del sistema que se construirá.
Salida	Alguna característica del sistema.	Descripción del sistema construido.
Tipos	CLASIFICACIÓN	DISEÑO
	ASESORAMIENTO	MODELADO
	DIAGNOSTICO	PLANIFICACIÓN
	MONITORIZACIÓN	ASIGNACIÓN
	PREDICCIÓN	SCHEDULING

[a]Entendemos por *sistema* un término abstracto que designa el objeto sobre el que se aplica la tarea. En diagnóstico técnico, por ejemplo sería el artefacto o aparato que está siendo diagnosticado.

Cuadro 4.2: Tareas Analíticas vs. Tareas Sintéticas.

4.3. Construcción de los modelos de Conocimiento

La metodología CommonKADS enfoca el modelo del conocimiento como un producto. Esto hace que se forme un cuello de botella por falta de experiencia en el modelado del conocimiento.

La solución, como es fácil de preveer, pasa por modelar, a su vez, el proceso.

No obstante, el modelado es una actividad constructiva para la que no existe una solución correcta ni un camino óptimo. Así pues, lo que se hace es proporcionar una guía que funciona bien en la práctica.

El modelado del conocimiento es una forma especializada de especificación de requisitos en el que se usan, por tanto, principios generales de la IS.

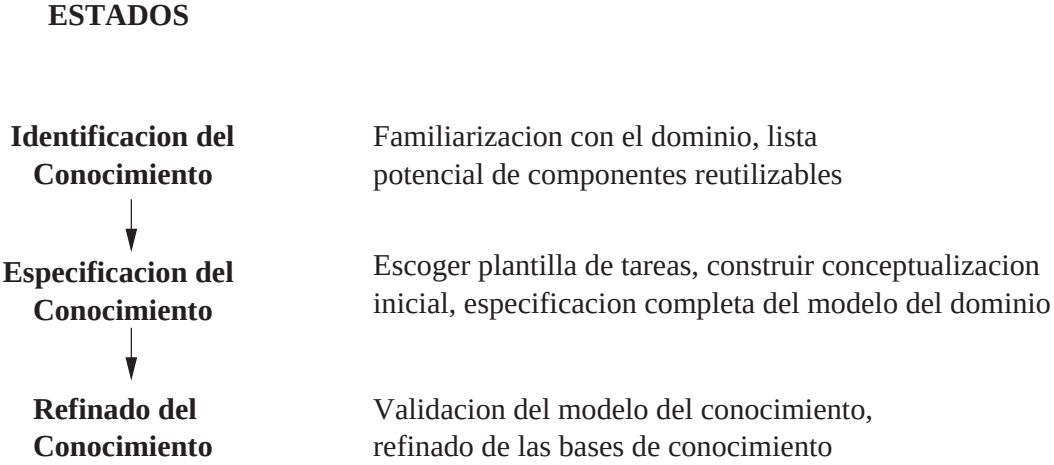


Figura 4.11: Guía para el modelado del Conocimiento.

4.3.1. Identificación del Conocimiento

META Estudiar los items de conocimiento, prepararlos para su especificación.

ENTRADA Tarea intensiva en conocimiento, principales items de conocimiento identificados, clasificación de la tarea de la aplicación.

ACTIVIDADES Explorar fuentes de información y estudiar la naturaleza de la tarea.

Los factores más importantes con respecto a las fuentes de información son su naturaleza (¿son claras? ¿tienen base teórica?) y su diversidad (¿son conflictivas? ¿con qué factor de riesgo?).

Las técnicas para su exploración son las tradicionales: marcado de textos, entrevistas, etc. El problema principal reside en encontrar un balance entre aprender sobre el dominio y convertirse en un experto.

Algunas guías:

- Hablar con la gente que trata a los expertos pero que no son expertos
- Evitar sumergirse en teorías complicadas y detalladas
- Construir unos cuantos escenarios típicos
- No pasar demasiado tiempo en esta actividad

Una vez acometidas estas actividades, puede realizarse una valoración de resultados, tanto tangibles (listado de fuentes, resúmenes de textos, glosario, descripción de escenarios), como intangibles (la propia comprensión).

La presencia de una lista de componentes tiene como objetivo allanar el camino en el manejo de componentes reutilizables en dos dimensiones:

- Dimensión de la Tarea (elegida del tipo asignado en el TM, construir una lista de plantillas)
- Dimensión del Dominio (tipo de dominio, buscar descripción estándar)

4.3.2. Especificación del Conocimiento

META Completar la especificación del conocimiento excepto para los contenidos de los modelos del dominio (que necesitan sólo contener instancias).

ACTIVIDADES

Elegir una plantilla de la tarea Como línea base de actuación, no debemos olvidar que existe una fuerte preferencia por un modelo de conocimiento basado en aplicaciones ya existentes (por razones de eficacia y calidad asegurada). Los criterios de selección son las características de la tarea de la aplicación (naturaleza de las entradas y salidas del sistema, restricciones del contexto...).

Algunas guías para la selección de plantillas:

- Preferir las que se usan con más frecuencia (por evidencia empírica).
- Construir una estructura inferencial anotada con ejemplos y un esquema del dominio.
- Si no se ajusta a ninguna plantilla, cuestionar la intensidad en conocimiento de la tarea.

Construir una conceptualización inicial del dominio Ha de construirse un esquema del dominio inicial, que tendrá dos partes:

- Conceptualización específica del dominio (que no es probable que cambie).
- Conceptualización de métodos específicos (sólo necesaria para resolver ciertos problemas —excepciones a la norma, no demasiado relevantes en este punto—).

Como "salida" de este paso obtendremos un esquema que debe cubrir al menos las conceptualizaciones específicas del dominio.

Algunas guías sobre cómo actuar:

- Utilizar en lo posible el modelo de datos existente (usar al menos la misma terminología en los constructos básicos; hará las cooperaciones e intercambios futuros más sencillos).
- Limitar el uso del lenguaje de modelo de conocimiento a conceptos, subtipos y relaciones (concentrarse en los "datos", de manera similar a cuando se construye un modelo de clases inicial).
- Si no existe modelo de datos disponible, usar técnicas estándar de IS para encontrar conceptos y relaciones.

Especificar las tres categorías del conocimiento Por último, se termina la especificación completa del dominio, pudiendo enfrentarse de dos maneras:

1. Ruta Middle-Out. Es la aproximación preferida, empieza con el conocimiento inferencial. Como precondición, la plantilla de la tarea ha de ser una buena aproximación de la estructura inferencial.
2. Ruta Middle-In. Comienza en paralelo con la descomposición de la tarea y el modelado del dominio, por lo que consume más tiempo. Es necesario si la plantilla de la tarea es de grano "demasiado grueso".

La estructura inferencial está suficientemente detallada si lo está la explicación que proporciona, o también si es fácil encontrar para cada inferencia un tipo de conocimiento del dominio que actúe tal y como se espera.

Algunas guías para especificar el *conocimiento de la tarea*:

- Empezar con la estructura de control.
- No incluir detalles de la memoria de trabajo.
- Elegir nombres de roles aclarativos (*Modelar es nombrar*).
- No incluir roles de conocimiento estático.

- En aplicaciones de tiempo real, considerar usar una representación alternativa al pseudocódigo (UML).

Algunas guías para especificar el *conocimiento inferencial*:

- Comenzar con la representación gráfica.
- Elegir los nombres de rol cuidadosamente (carácter dinámico, hipótesis, datos iniciales,...).
- Usar un conjunto lo más estándar posible de inferencias.

Algunas guías para especificar el *conocimiento del dominio*:

- Usar como roles estáticos los tipos del dominio (no tienen que tener la representación correcta, esta será una tarea del diseño).

4.3.3. Refinado del Conocimiento

El refinado del Conocimiento pasa por:

1. Validación del modelo de Conocimiento.
2. Rellenar las Bases de Conocimiento.

Validación del modelo de Conocimiento

La *validación* debe ser **interna** (verificación, ¿es el modelo adecuado?) y **externa** (validación contra los requisitos del usuario, ¿es correcto el modelo?).

Existen diferentes técnicas de validación:

- Interna:
  - Rutas estructuradas (probar escenarios típicos)
  - Herramientas software
- Externa:
  - Suele ser más difícil y/o amplia
  - Técnica principal: simulación (prototipos, simulación basada en papel)

En cuanto al mantenimiento, según el punto de vista de CommonKADS, no es algo diferente del desarrollo del modelo, pues es algo cíclico. El modelo es como un repositorio de información; debemos especificar requisitos para obtener herramientas de soporte potentes.

Rellenar las Bases de Conocimiento

El esquema contiene dos tipos de dominios: tipos de información parte de un caso y tipos de información parte de un modelo de conocimiento.

La meta es determinar todas las instancias de cada tipo. Las instancias sólo se necesitan para un escenario.

Algunas guías para el rellenado de las bases de conocimiento:

- Rellenar las bases de conocimiento es una forma de validar el esquema construido.
- Normalmente no es posible definir una base de conocimientos correcta y completa en un primer ciclo.
- Las bases de conocimiento necesitan mantenimiento (el conocimiento cambia con el tiempo).
- Técnicas: incorporar facilidades de edición para las bases de conocimiento, trazas, entrevistas estructuradas, aprendizaje automático, mapeado de otras bases de conocimiento.

4.3.4. Documentación del modelo de Conocimiento

Una vez construido el modelo, se redacta un documento KM-1 (ver apéndices), que contendrá:

- Especificación del modelo de conocimiento.
- Listado de fuentes de información.
- Listado de los componentes reusables del modelo.
- Escenarios típicos del problema que resuelve la aplicación.
- Resultados de validación de la simulación.
- Material de elicitación.

4.4. El modelo de Comunicación

El papel del **modelo de Comunicación** es especificar los procesos de transferencia de información/conocimiento. Es, en cierto modo, un control de nivel superior sobre la ejecución de la tarea (múltiples tareas intensivas en conocimiento). Tareas de comunicación adicionales, pueden, además, añadir facilidades de explicación al sistema. Un ejemplo es la interación básica sistema-usuario.

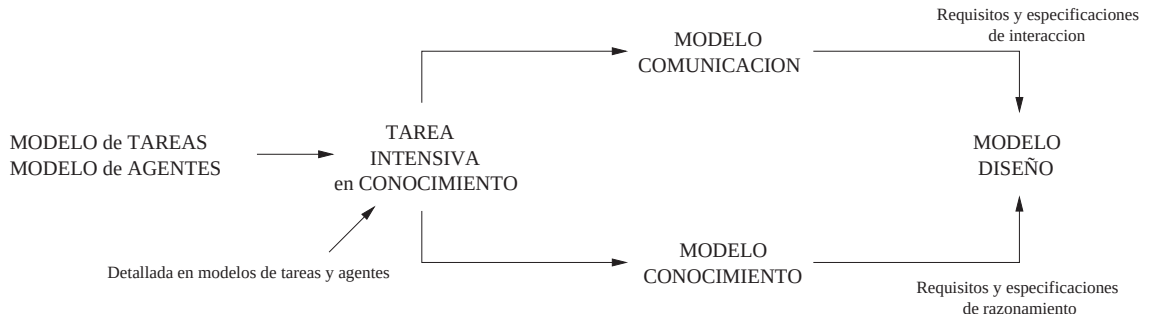


Figura 4.12: Relación del modelo de Comunicación con otros modelos.

Las "entradas" al modelo de Comunicación son:

Modelo de Tareas Lista de tareas *hoja* llevadas a cabo por los agentes considerados.

Modelo de Conocimiento Funciones de transferencia.

Modelo de Agentes Descripción de agentes relevantes, capacidades, responsabilidades y restricciones.

Cada vez más, los sistemas de información son información + sistema de comunicación:

- √ Aplicaciones distribuidas (telemática).
- √ Organismos virtuales.
- √ Sistemas multiagente inteligentes.
- √ Manejo de flujos de trabajo.
- √ Ingeniería concurrente.
- √ Manejo e integración de la cadena de negocio.

El modelado de la información debe cubrir:

- Análisis de la organización.
- Análisis de las tareas.
- Análisis de actores/agentes (sistemas y humanos).

Normalmente, varios actores cooperan en un proceso de negocio o tarea. El modelo de Comunicación se centra en modelar el diálogo entre agentes afrontándolo mediante una aproximación semiformal estructurada.

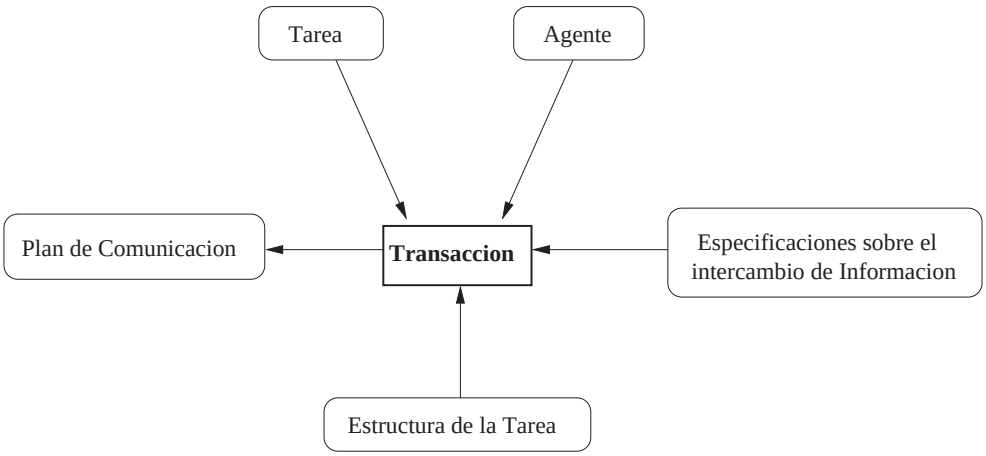


Figura 4.13: Estructura del modelo de Comunicación.

La aproximación por capas al modelado de las comunicaciones consta de tres niveles:

Plan de Comunicaciones general.- Gobierna el diálogo completo entre dos agentes.

Transacciones individuales.- Son las que unen dos tareas *hoja* llevadas a cabo por dos agentes diferentes.

Especificación del intercambio de información.- Detalla la estructura de las transacciones.

Como se puede ver, pues, las **transacciones** son el componente clave del modelo de Comunicación. Describen qué objetos de información se intercambian, indicando los agentes y tareas implicados. Son el bloque de construcción para el diálogo completo entre un par de agentes, y tienen una estructura interna.

Haciendo abuso de lenguaje, suele llamarse transacción incluso a lo que se intercambia entre dos tareas llevadas a cabo por diferentes agentes.

A un nivel superior, está el **plan de comunicación**, que gobierna el diálogo completo entre los agentes, siendo la especificación concreta del modelo de Comunicación.

4.4.1. Plan de Comunicación

Generalmente es más fácil comenzar por el plan de comunicación global. El plan de comunicación describe completamente el diálogo de alto nivel, siendo sus transacciones típicas: entrada de datos, contestación de preguntas, presentación de resultados, etc.

Actividades

Para cada agente se confeccionará una lista de todas las tareas en las que participa, y para cada tarea se identificará el conjunto de transacciones agente-agente asociadas.

El resultado se combina en un *diagrama de diálogo* (DD, ver figura 4.14) que representa las transacciones entre cada par de agentes que se comunican. Se dibuja, pues, un DD para cada combinación de dos agentes que intercambian información, especificando de esta manera el control sobre las transacciones.

Como alternativa a la notación del DD, se puede utilizar también pseudocódigo con primitivas de control especiales: ENVIAR, RECIBIR, LLEVAR_A_CABO, ESPERAR, PROCESAR, REPETIR,...

4.4.2. Transaciones agente-agente

El nivel de especificación medio del modelo de Comunicación está encarnado en la especificación de las transacciones individuales, estructuradas en un número de componentes.

Técnicas simples de formulario son útiles aquí (ver formulario CM-1 en apéndices).

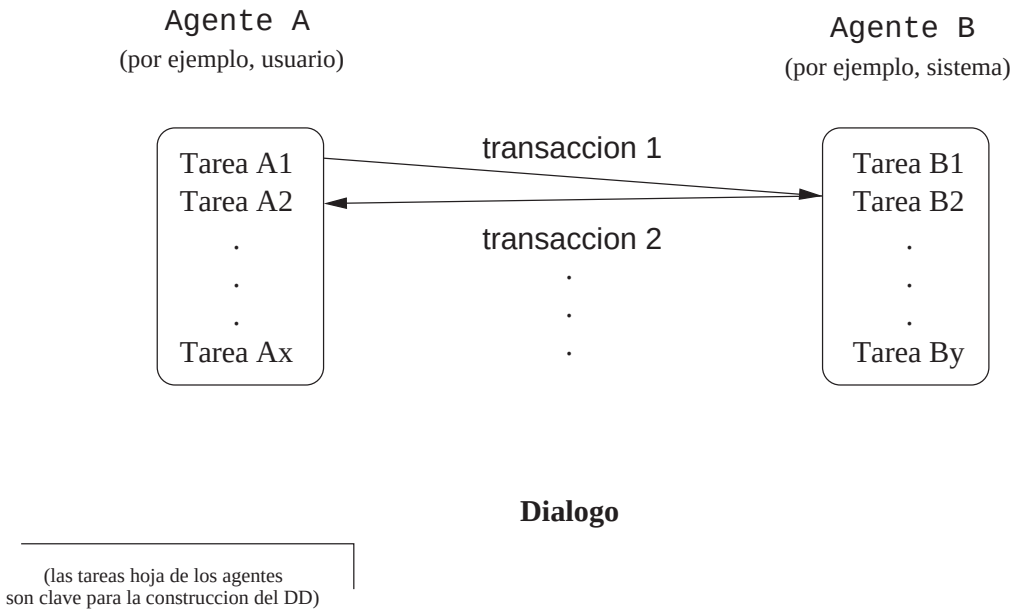


Figura 4.14: Estructura general de un Diagrama de Diálogo.

Las transacciones suelen agruparse tras un único plan de comunicación, salvo en sistemas multiagente.

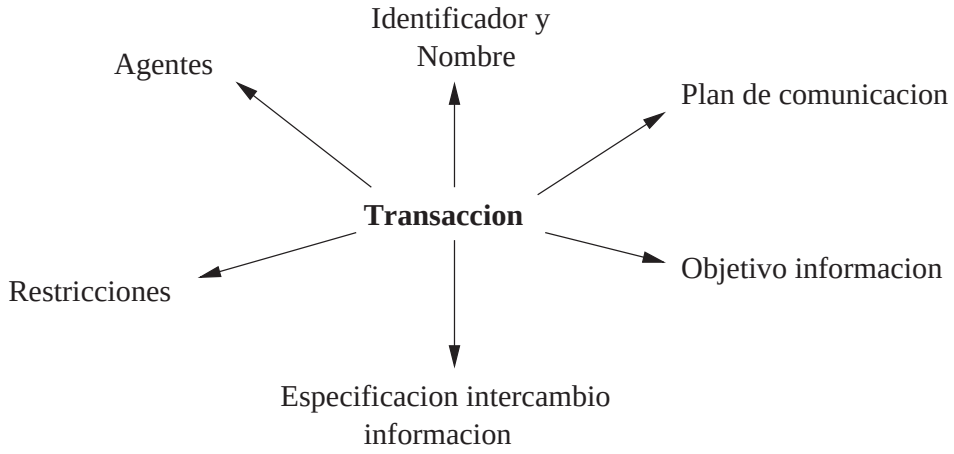


Figura 4.15: Esquema de la estructura de una transacción (CM-1).

Tareas Delegación	Tareas Adopción	Tareas Intercambio
Request	Propose	Ask
Require	Offer	Reply
Order	Agree	Report
Reject_td	Reject_ta	Inform

Cuadro 4.3: Tipos de comunicación.

Las transacciones tienen, en general, una primera parte informativa y una segunda parte de "solicitud de acción" (usualmente un mensaje de delegación de tarea). Las transacciones no sólo admiten, pues, un contenido, sino también una relación pretendida entre dos agentes. Ambos aspectos deben especificarse explícitamente.

Los lenguajes de comunicación de agentes (ACL) están inspirados a menudo por la teoría del *acto del habla*, que hace distinciones entre el contenido y el efecto pretendido.

4.4.3. Patrones transaccionales

Ya por último, nos centraremos en el nivel de detalle del modelo de Comunicación, que consiste en la especificación detallada del mensaje:

- ⋆ Su contenido, expresado mediante una declaración proposicional (locución).
- ⋆ Su intención[2], expresada mediante un mensaje escrito (ilocución).

Los tipos predefinidos son los ya indicados en la tabla 4.3 (SOLICITAR, EXIGIR, ORDENAR, RECHAZAR; PROPONER, OFRECER, ACORDAR, RECHAZAR; PREGUNTAR, RESPONDER, INFORMAR y ENVIAR_INFORME).

Cuadro 4.4: Semántica de algunos tipos de comunicación.

REQUEST/PROPOSE	Negociación para colaborar
REQUIRE/OFFER	Compromiso condicional
ORDER/AGREE	Efectuar un acuerdo
REJECT	Negarse a efectuar la petición
ASK/REPLY	Preguntar sobre información y recibir respuesta
REPORT	Informe como consecuencia de un acuerdo previo
INFORM	Acción informativa independiente

Por supuesto, no sólo es posible enviar mensajes simples, también se pueden formar cadenas naturales de tipos de mensajes.

El resumen de la especificación de los intercambios de información se aglutina en el formulario CM-2, que sólo suele ser necesario para patrones de comunicación complejos (es un detalle del CM-1). Su estructura es la siguiente:

- ◦ Identificador y nombre de la transacción.
- ◦ Agentes involucrados (emisor, receptor).
- ◦ Items de información; se componen a su vez de:
  - rol (objeto central + item soporte[3])

[2] Intención = propósito + cometido.

[3] Textos explicativos de material del dominio, trazados de razonamiento, explicaciones porqué/como.

  - forma sintáctica (cadenas de datos, diagramas, etc.)
  - medio (ventana pop-up, interfaz de línea de comandos, intervención humana...)
- Especificación del mensaje.
- Control del mensaje (es un refinado del control especificado en el plan de comunicación, que usará la misma notación: diagramas de estado o pseudocódigo).

4.4.4. Técnicas de validación

Para validar el modelo de Comunicación suelen emplearse *walkthroughs* en el plan de comunicación, para verificar la adecuación de la estructura de las transacciones, la completitud de la lista de items de información y la necesidad de ayuda o explicación.

Existen técnicas más formales, como la técnica **Mago de Oz**, que se basa en la misma idea que el test de Turing. No obstante, su uso es caro pues es una técnica experimental para validar la interacción que requiere de la construcción de un software maqueta.

Guía de Nielsen para la usabilidad

- Presentar diagramas simples y naturales.
- Hablar el lenguaje del usuario.
- Minimizar la carga de memoria del usuario.
- Mantener la consistencia de la terminología.
- Proporcionar información sobre lo que está pasando (retroalimentación).
- Mostrar salidas claramente marcadas desde los estados no deseados.
- Ofrecer atajos al usuario experto.

Guías para pesar el modelo de Comunicación

- Entradas clave:
  - ↪ Tareas hoja del TM.
  - ↪ Funciones de transferencia del KM.
- Tener en cuenta las capacidades de los agentes (AM).
- La formulación sintáctica del medio es algo entre el CM y el DM (se aborda en el CM si existen razones conceptuales para ello).
- Decidir sobre la información de soporte (no en DM).

Actividades del modelo de Comunicación

- Identificar los objetos de información centrales para ser intercambiados entre agentes.
- Identificar las transacciones asociadas.
- Dibujar los DD importantes.
- Combinar esto en un plan de comunicaciones completo.
- Especificar las transacciones individuales (CM-1 y CM-2).
- Validar y pesar el modelo.

Capítulo 5

Del análisis a la implementación: el modelo de Diseño en CommonKADS

El *Diseño* del sistema recibe como entradas:

- √ El modelo de Conocimiento (requisitos para la resolución de problemas)
- √ El modelo de Comunicación (reglas de interacción)
- √ Otros modelos (requisitos "no funcionales")

Y obtendrá como salidas la especificación de una arquitectura software y el diseño de la aplicación dentro de dicha arquitectura.

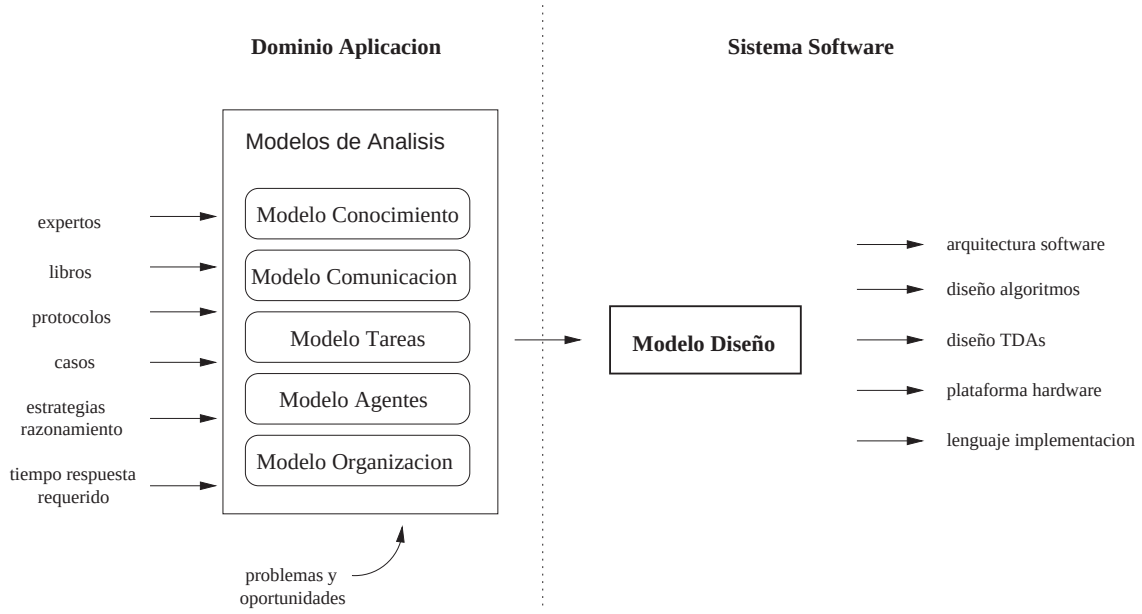


Figura 5.1: Del análisis al diseño en CommonKADS.

Entendemos por *arquitectura* del sistema la descripción del software en términos de descomposición en subsistemas, seleción del régimen(es) de control y descomposición de los subsistemas en módulos software.

Especificar esta arquitectura es el punto central el proceso de diseño, y se parte de una serie de arquitecturas de referencia para sistemas basados en CommonKADS.

5.1. Principio de Conservación de la Estructura

El **principio de Conservación de la Estructura** es el principio central del diseño moderno:

> "*Debe preservarse el contenido y la estructura del modelo de análisis durante el diseño.*"

Según esta filosofía, diseñar es "añadir detalles específicos de implementación a los resultados del análisis", preservando la información como noción clave.

Esto está directamente relacionado con los criterios de calidad del diseño en general:

- √ Minimizar el acoplamiento.
- √ Maximizar la cohesión.
- √ Transparencia.
- √ Mantenimiento.

A estos criterios generales, cuando hablamos de diseño de SBCs, se añaden los siguientes:

- √ Reusabilidad de elementos de diseño/código resultante.
- √ Mantenimiento y adaptabilidad (el desarrollo en un solo paso es normalmente poco realista, especialmente para sistemas intensivos en conocimiento).
- √ Potencia explicativa.
- √ Adquisición de conocimiento/facilidad para el refinado (el conocimiento cambia con el tiempo).

5.2. El modelo de Diseño

En la construcción del **modelo de Diseño** se seguirán los siguientes pasos:

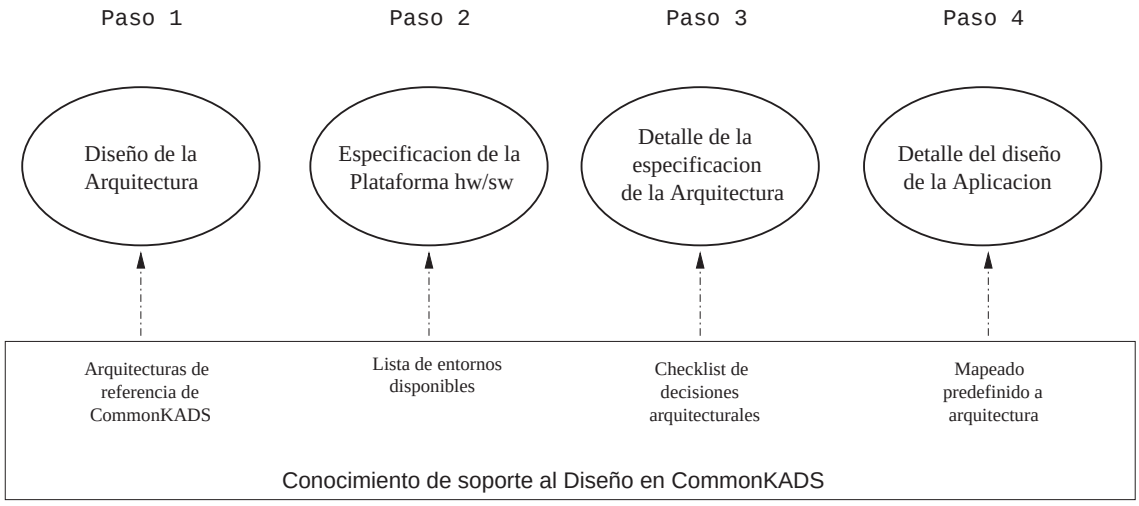


Figura 5.2: Pasos en la construcción del modelo de Diseño.

5.2.1. Diseño de la arquitectura del sistema

El primer paso en la construcción del modelo de Diseño es, pues, la especificación de la arquitectura global.

El principio que se sigue es separar la funcionalidad de aspectos de la interfaz, para lo que, como ya se ha mencionado, se descompone el sistema en subsistemas, se define un régimen de control global y se descomponen los subsistemas en módulos software.

La arquitectura global seguirá el **modelo MVC** (*Model View Controller*), que fue desarrollado para el diseño O.O. en lenguaje Smalltalk-80 pero que ha sido adoptado mayoritariamente en el diseño de software. Este modelo distingue entre los objetos de una aplicación y su visualización y define una unidad de control central con régimen dirigido por eventos. El sistema construido siguiendo esta filosofía tendrá tres subsistemas principales, indicados en la figura 5.3.

Subsistema Modelo de Aplicacion

El modelo de la aplicación contiene los datos y funciones de la aplicación, esto es, los objetos del modelo de conocimiento.

Los datos están presentes en forma de bases de conocimiento y datos dinámicos manipulados durante el razonamiento (por ejemplo, roles dinámicos), mientras que las funciones están representadas por las tares, inferencias y funciones de transferencia.

Subsistema Vistas

Este subsistema se encarga de la visualización de los datos y funciones de la aplicación, haciendo posible la visualización de la información estática y dinámica a los agentes externos, como el usuario o bien otro sistema software.

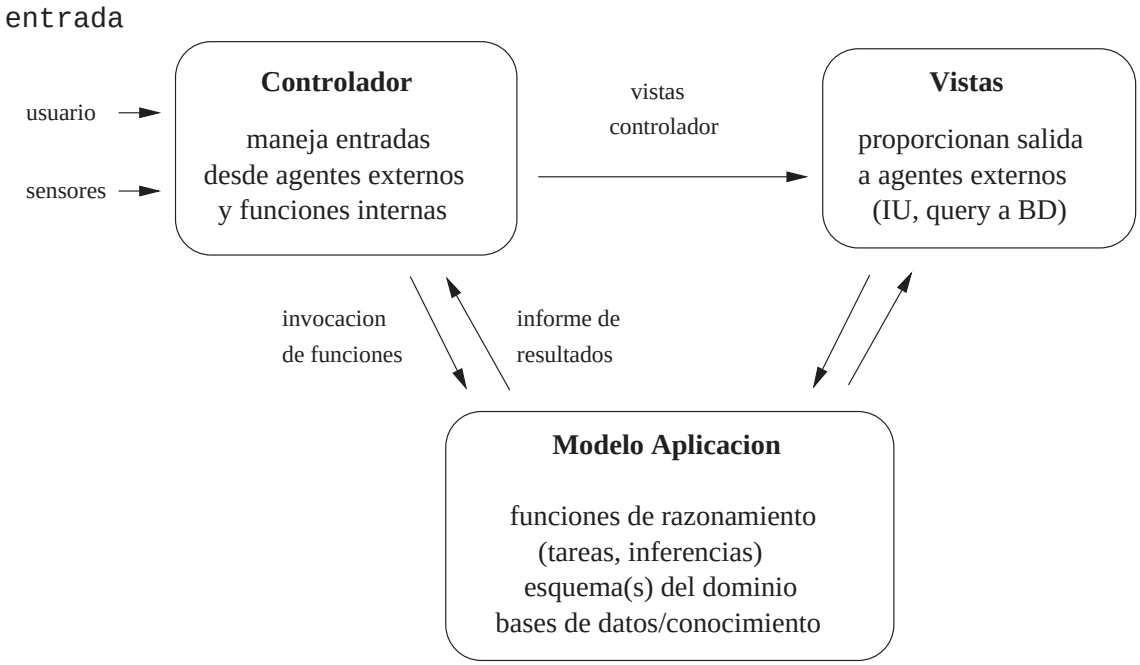


Figura 5.3: Esquema del *Model View Controller*.

Es posible tener visualizaciones múltiples, agregar la visualización de múltiples objetos de la aplicación, etc.

La inclusión de este subsistema requiere actualización arquitectural o bien mecanismos de integración, como pueden ser una tabla de mapeos o protocolos de mensajes para notificación de cambios en el estado de los objetos.

Subsistema Controlador

Es la unidad central de control y comandos. Suele estar dirigido por eventos, proporcionando *handlers* para eventos tanto externos como internos.

Permite la activación de las funciones de la aplicación y decide qué hacer cuando llegan los resultados. Puede definir sus propias *vistas de control* para proporcionar información sobre el proceso (por ejemplo, al usuario experto). Suele tener un reloj interno y una agenda, pudiendo tener un comportamiento tipo *daemon*.

Algunos otros aspectos importantes de la arquitectura MVC, además de haber sido desarrollada en el contexto de la O.O. (de hecho, es una descomposición funcional de "objetos"), pasan por darnos cuenta de que su uso no está necesariamente restringido a una aproximación al diseño/implementación O.O., aunque el paradigma de paso de mensajes debe ajustarse bien a los entes de la arquitectura.

A la hora de descomponer el modelo de la aplicación en subsistemas debemos tener en cuenta que se debe permitir el diseño preservando la estructura, así como permitir la integración con otras aproximaciones de la IS.

Las opciones son dos: una descomposición funcional o bien una descomposición O.O. La opción escogida por CommonKADS es la segunda, ya que se ajusta bien al carácter declarativo de las especificaciones de los objetos en el modelo de conocimiento (puede verse una tarea como un objeto) y además simplifica el mapeado con implementaciones O.O. en muchas herramientas.

Este primer paso en la construcción del modelo de Diseño queda reflejado en el modelo DM-1 (ver apéndices).

5.2.2. Identificación de la plataforma de implementación

El segundo paso en la construcción del modelo de Diseño es la identificación de la plataforma de implementación.

Los requisitos específicos del cliente suelen restringir esta selección, lo que es una razón para colocarla en un paso temprano dentro del proceso. Hoy en día, la selección del software es mucho más importante que la del hardware, aunque puede no serlo en el caso de aplicaciones en tiempo real.

En caso de que la selección fuese más o menos libre, deberíamos considerar posponerla hasta que se finalizase el tercer paso (especificación de los componentes de la arquitectura).

Algunos criterios para la selección de la plataforma de implementación pueden ser:

- ▷ Existencia de librerías de clases de objetos "vista" (puede ser necesario construir muchos uno mismo en caso contrario).
- ▷ Formalismo en la representación del conocimiento (preferentemente una representación declarativa).
- ▷ Existencia de interfaces estándar con otro software (ODBC, CORBA,... suelen ser necesarias a menudo).
- ▷ Facilidades para la escritura del lenguaje (un "tecleado débil" normalmente implica más trabajo en el mapeado del modelo de análisis).
- ▷ Facilidades de control/protocolos (soporte de paso de mensajes, posibilidad de multi-threading).
- ▷ Soporte de CommonKADS (extensión de plataformas dedicadas — por ejemplo, librerías de objetos—, links con herramientas CASE que soporten CommonKADS).

Este segundo paso en la construcción del modelo de Diseño queda reflejado en el modelo DM-2 (ver apéndices).

5.2.3. Especificación de los componentes de la arquitectura

El tercer paso en la construcción del modelo de Diseño es la especificación de los componentes arquitecturales.

Esta especificación consiste en definir los componentes de la arquitectura en más detalle. En particular, se definen las interfaces entre los subsistemas y/o módulos de los sistemas, especificando sus componentes.

Se realiza así un diseño general de las utilidades de la arquitectura. Algunas plataformas incorporan una arquitectura CommonKADS en las que las decisiones han sido predefinidas; esto tiene como ventaja que este paso se hace más rápido y fácil, pero por contra destruye la capacidad creativa.

Utilidades del Controlador

El controlador realiza un control dirigido por eventos con un componente central de control. Puede verse como una implementación del modelo de comunicación.

Las declaraciones típicas son:

* Activación y finalización de funciones de la aplicación.
* Decisión sobre la posibilidad de que el usuario realice interrupciones para informarse del trazado/contexto.
* Posibilidad de abortar funciones.
* Manejo de funciones de transferencia/transacciones.
* Necesidad o no de procesado concurrente.

Utilidades del Modelo de Aplicación

* Tarea: para los objetos necesitamos definir dos operaciones:
  - Un método de inicialización, para iniciar los valores de la tarea.
  - Un método de ejecución, que invoque al método de la tarea.
* Método de la Tarea:
  - Elementos del lenguaje de control (constructos de control).
  - Declaratividad del lenguaje de control (por ejemplo, en O.O., la implementación natural es una operación EXECUTE, pero destruye la naturaleza declarativa; es necesario "objetificar" la estructura de control).
* Inferencias:
  - Tres operaciones principales: EXECUTE, MORE_SOLUTIONS y HAS_SOLUTION.
  - Enlaces a los métodos de inferencia.
* Métodos de Inferencia:

    - Librería de métodos.
    - Permitir relaciones muchos-a-muchos entre métodos e inferencias.
  - Funciones de transferencia:
    - Implementación vía patrones de paso de mensajes.
  - Roles dinámicos:
    - Tipos de datos permitidos: ELEMENTO, CONJUNTO, LISTA,...
    - Operaciones de acceso/actualización, selección, eliminación, añadir, etc.
  - Roles estáticos:
    - Funciones de acceso/consulta.
  - Modelo del dominio (bases de conocimiento):
    - Formato representacional.
    - Funciones de acceso/consulta.
    - Funciones de modificación/análisis.
  - Constructos del dominio:
    - Simplemente inspeccionar.

Utilidades de las Vistas

  - Visualizaciones gráficas estándar.
  - Generación de formatos externos (por ejemplo, consultas SQL).
  - Utilidades arquitecturales de actualización de vistas:
    - Tablas de mapeado.
    - Protocolos de mensajes.

Interfaces de Usuario

  - Interfaz con el usuario normal:
    - Considerar utilidades especiales (por ejemplo, generación de lenguaje natural).
    - Uso de visualizaciones específicas del dominio (depende del diseño de la aplicación).
  - Interfaz con usuarios expertos:
    - Interfaz de trazados.
    - Interfaz de edición/refinado para bases de conocimiento.

Este tercer paso en la construcción del modelo de Diseño queda reflejado en el modelo DM-3 (ver apéndices).

5.2.4. Especificación de la aplicación en el contexto de la arquitectura

El último paso en la construcción del modelo de Diseño es la especificación del diseño en el contexto de la arquitectura. Esta tarea se divide en dos:

1. *Mapear la información de análisis en la arquitectura.*- Se necesitan construir o disponer de herramientas de mapeo (por ejemplo, un API). La extensión del mapeo depende de las decisiones que están ya construidas en la arquitectura.
2. *Añadir detalles de diseño.*- Debe hacerse el diseño de la aplicación para el controlador:
   - Su entrada principal es el Modelo de Comunicación.
   - A menudo es necesario el procedimiento natural.
   - Como mínimo es necesario un procedimiento de bootstraping.
   - Otras funciones:
     - manejo de requisitos de explicación
     - control de usuario sobre el proceso de razonamiento
     - interrupción del razonamiento/control estratégico
     - permitir escenarios *what-if* (simulación)

La especificación de la aplicación en el contexto de la arquitectura queda reflejado en el formulario DM-4 (ver apéndices).

5.3. Diseño de prototipos

El "prototipado rápido" tiene una mala reputación porque este término ha sido empleado para referirse a implementaciones rápidas y poco cuidadas imposibles de mantener y escalar. No obstante, hoy en día está considerada como una buena técnica para comprobar el grado de comprensión que se tiene sobre aquello que se desea llegar a contruir.

5.3.1. Prototipado de subsistemas de razonamiento

¿Cuándo puede ser necesario construir un prototipo del subsistema de razonamiento?

- Cuando contemos con elementos recientemente construidos en el modelo de conocimiento (plantillas nuevas).
- Cuando sospechemos de agujeros en el conocimiento del dominio.
- En general, para validar y verificar el modelo del dominio:
  - validar (¿es el sistema adecuado?)

- verificar (¿es adecuado el sistema?)

Además la plataforma de implementación debe soportar el prototipado, su construcción debe ser cuestión de días.

5.3.2. Prototipado de interfaces de usuario

La construcción de un prototipo de interfaz con el usuario nos brinda la oportunidad de comprobar la interfaz sin necesidad de desarrollar toda su funcionalidad. Puede ser necesario si el formato de vistas es complejo e incluso para poder construir una interfaz externa más completa.

5.4. SBCs distribuidos

Hay varios campos en los que potencialmente sería interesante usar una arquitectura distribuida a la hora de construir un SBC:

Servicios de razonamiento
: El modelo de aplicación funciona como un servicio. No es necesaria una interfaz de usuario.

Servidores basados en conocimiento/ontológicos
: Unifican la terminología SSEE. Un ejemplo sería el GRASP, un servidor para piezas de arte.

Servicio de métodos
: Un sistema distribuido en forma de un conjunto de métodos.

Y, por supuesto, combinaciones de los mencionados.

Capítulo 6

Técnicas para la adquisición del conocimiento

Llamamos **adquisición del conocimiento** al proceso de recoger los datos e información que necesitamos para construir nuestro SBC.

Para ello se utilizan una serie de técnicas que pueden ser *manuales*, usualmente más fáciles de usar por parte del ingeniero de conocimiento y más fáciles de aceptar por el experto aunque más costosas en tiempo, *semiautomáticas* e incluso completamente *automáticas*.

El problema es que la adquisición del conocimiento no es una actividad inmediata. Los propios expertos no son conscientes de su conocimiento y su forma de actuar, que les resulta complicado verbalizar, pues muchas veces es pragmático. Para paliar esto en mayor o menor medida es útil elegir, dentro de lo posible, varios expertos, que pueden ser de distintos tipos:

Experto teórico o académico: posee conocimientos más encaminados a la parte docente, más formales. Suele ser capaz de explicar racionalmente pero está poco relacionado en la práctica.

Experto práctico: resuelve los problemas día a día, aporta una visión más "real" (escenarios). Da soluciones, es un experto técnico, aplica algo porque funciona, sin quizás tener una explicación formal.

Pese a todo, nunca debemos olvidar que existe un componente personal muy importante, que no se puede evitar.

6.1. Escenarios de adquisición del conocimiento

Hay cuatro escenarios típicos de adquisición del conocimiento:

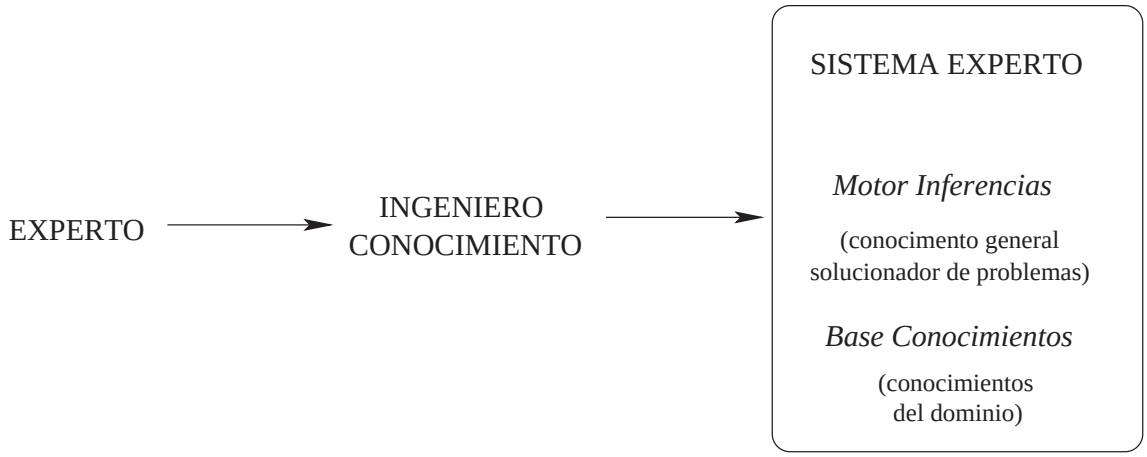


Figura 6.1: Primer escenario de adquisición del conocimiento.

El escenario 1 (figura 6.1) es el más típico.

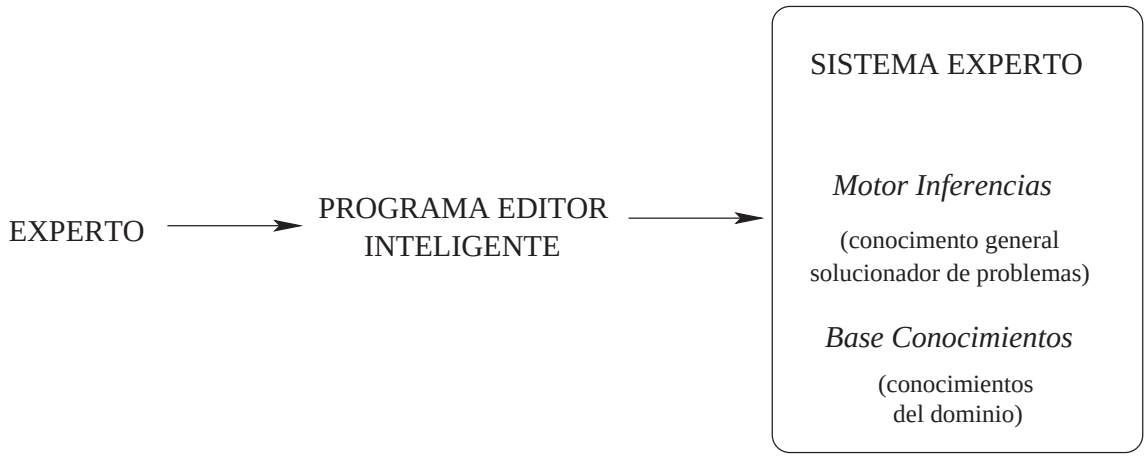


Figura 6.2: Segundo escenario de adquisición del conocimiento.

El escenario 2 (figura 6.2) surge de la mente de McCarthy (1968) y su "Advice Taker", aunque su mejor trabajo fue TEIRESIAS (1976). Consiste en que el experto *habla* con el programa en lugar de con el ingeniero de conocimiento (¡que es, obviamente, el que ha construido el programa!), ayudando de este modo a que se implique y favoreciendo la depuración. Por supuesto, lo extremadamente complejo es la construcción del mencionado sistema/programa interlocutor.

El programa de inducción —por ejemplo, una red de neuronas— (escenario 3, figura 6.3) intenta extraer algún tipo de conocimiento, generalizaciones fundamentalmente, a partir de los datos. Este conocimiento se traducirá posteriormente al "mundo" del SBC (paso que, por cierto, puede ser no trivial ni directo). La ventaja es que ya es una técnica automática (salvo en la introducción de los datos). El problema, que la construcción de estos programas de inducción que generalicen algo útil no es fácil.

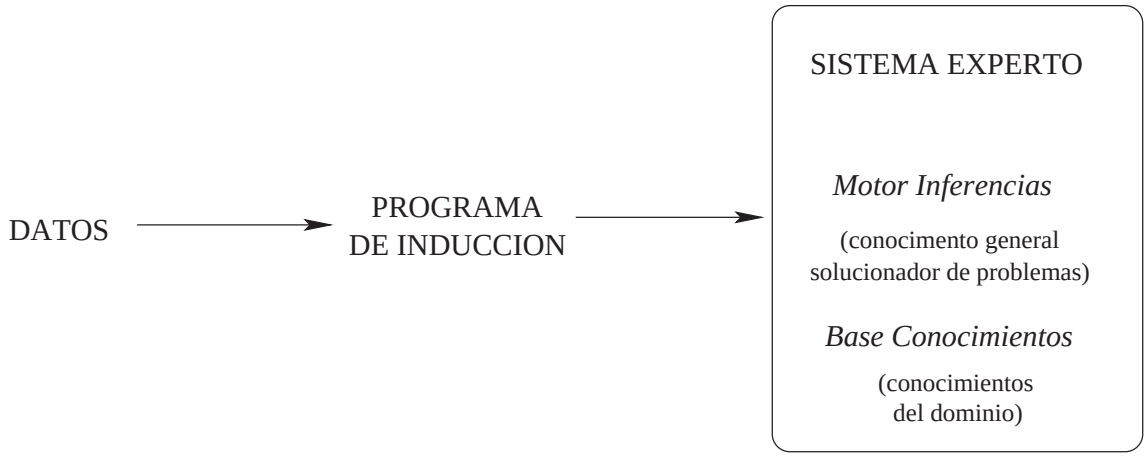


Figura 6.3: Tercer escenario de adquisición del conocimiento.

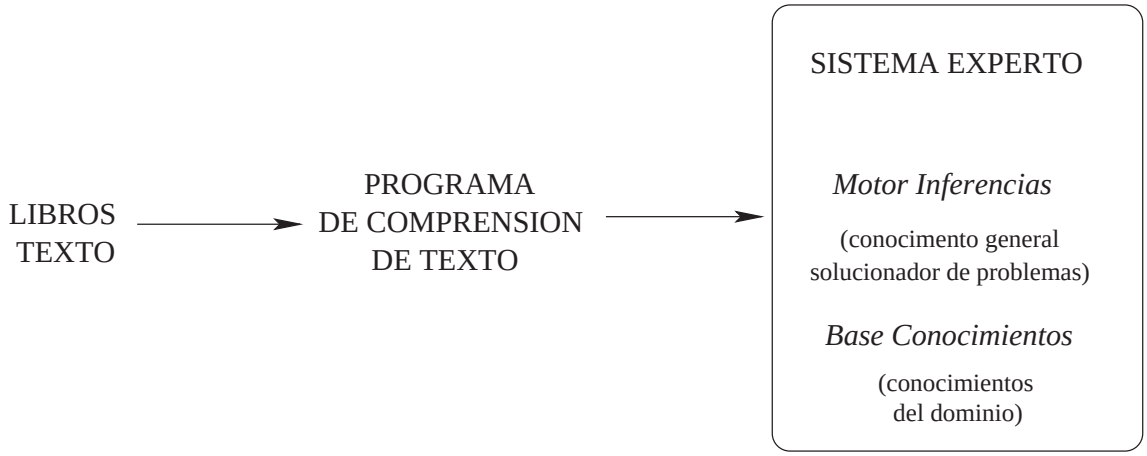
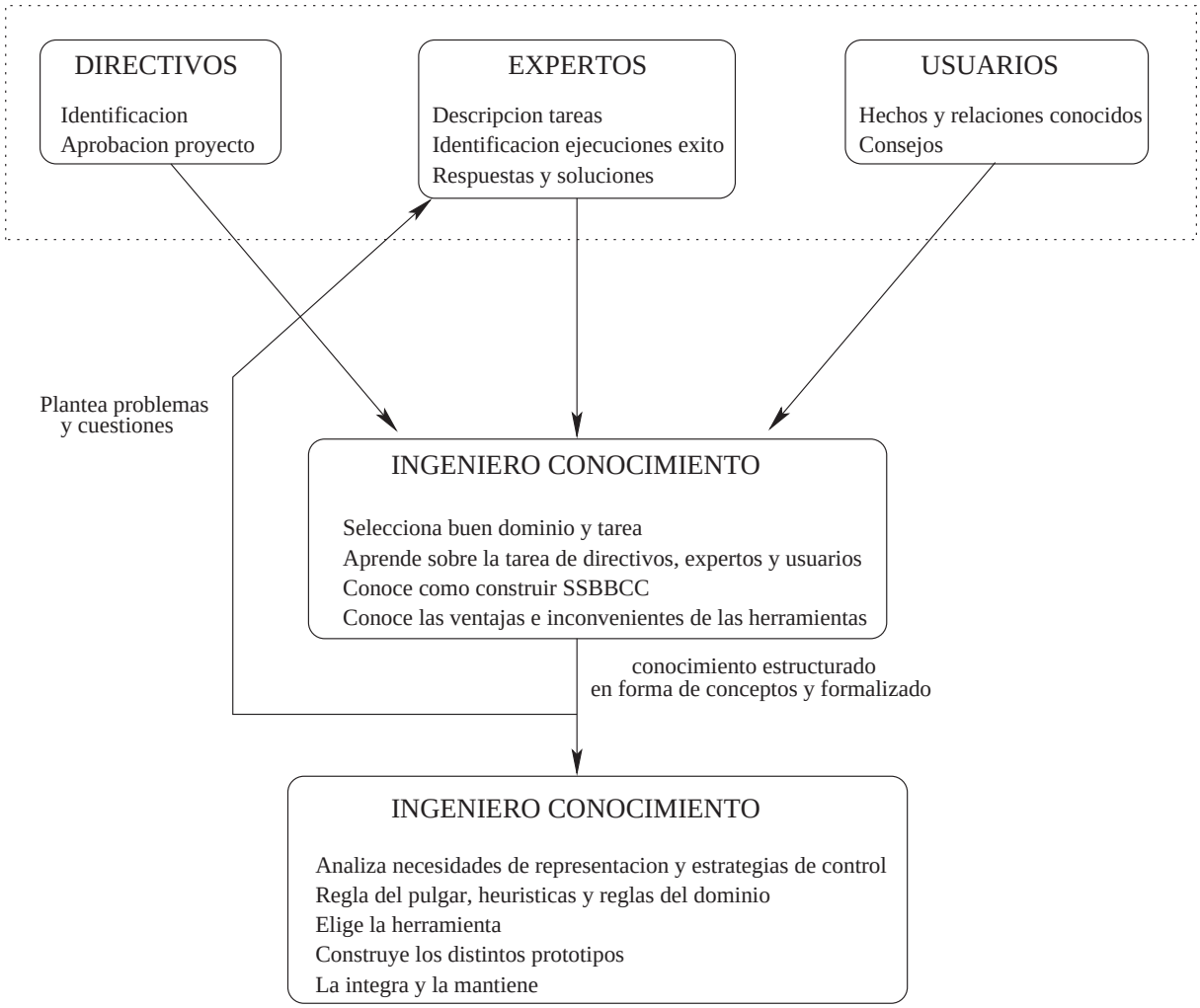


Figura 6.4: Cuarto escenario de adquisición del conocimiento.

El programa de comprensión de texto (escenario 4, figura 6.4) debería comprender diagramas, esquemas, y poseer un "criterio". La interpretación del lenguaje natural, aunque sea referido a un campo específico, es algo muy complicado.

El conocimiento, por su parte, también puede ser de muchos tipos (lo iremos viendo).


DIRECTIVOS
Identificacion
Aprobacion proyecto
EXPERTOS
Descripcion tareas
Identificacion ejecuciones exito
Respuestas y soluciones
USUARIOS
Hechos y relaciones conocidos
Consejos
Plantea problemas
y cuestiones
INGENIERO CONOCIMIENTO
Selecciona buen dominio y tarea
Aprende sobre la tarea de directivos, expertos y usuarios
Conoce como construir SSBBCC
Conoce las ventajas e inconvenientes de las herramientas
conocimiento estructurado
en forma de conceptos y formalizado
INGENIERO CONOCIMIENTO
Analiza necesidades de representacion y estrategias de control
Regla del pulgar, heuristicas y reglas del dominio
Elige la herramienta
Construye los distintos prototipos
La integra y la mantiene

Decálogo del Ingeniero de Conocimiento[a]

Facultades de Comunicación

- Utilización efectiva del lenguaje (oral y escrito).
- Capacidad de representación esquemática.
- Capacidad de interpretación.
- Trato agradable.

Inteligencia

- Capacidad de aprendizaje.
- Apertura de mente y flexibilidad.

Tacto y Diplomacia

- Reflexión y tacto.

Energía y Paciencia

- Valoración del trabajo en equipo.
- Capacidad de decisión, discusión, crítica y estimulación.

Persistencia

Lógica

- Claridad de pensamiento, capacidad de orden.

Versatilidad e Inventiva

- Potencia analítica.
- Imaginación.

Autoconfianza

Conocimiento del Dominio de Aplicación

Conocimiento acerca de la Programación del Sistema

[a]El problema es que estas todas estas cualidades no suele reunirlas una sola persona, de modo que es necesario la creación de grupos interprofesionales.

Cuadro 6.1: Decálogo del Ingeniero de Conocimiento.

Tipo Conocimiento	Actividad	Técnica
DECLARATIVO	Búsqueda de heurísticas generales	Entrevistas no estructuradas
PROCESAL	Búsqueda de rutinas y procedimientos	Entrevistas estructuradas
SEMÁNTICO	Búsqueda de conceptos y vocabulario	Observación directa Análisis de Tareas
	Heurísticas y Procedimientos de toma de decisiones	Emparrillado Clasificaciones Trazado del proceso de razonamiento
EPISÓDICO	Búsqueda de heurísticas analógicas de solución de problemas	Simulaciones Trazado del proceso de razonamiento (Análisis de protocolos)

Cuadro 6.2: Métodos de Adquisición del Conocimiento.

La técnica a utilizar para la adquisición del conocimiento depende no sólo del tipo de conocimiento a adquirir, sino también del dominio, circunstancias particulares, etc.

6.2. Las entrevistas

Las **entrevistas** son un método sencillo, manual, de adquisición del conocimiento que puede ser utilizado por cualquier persona, en cualquier dominio, con cualquier tipo de experto y con cualquier tipo de conocimiento.

Las primeras entrevistas suelen recibir el nombre de *entrevistas de despliegue* y su finalidad es interaccionar con el experto, conocerse mutuamente, etc. Normalmente hay que vencer una mala predisposición, explicar lo que se pretende hacer, lo que se espera del experto,... y por tanto su duración no debería ser muy extensa.

A continuación aparecen las *sesiones de adquisición* que son de dos tipos:

- No estructuradas (sesiones de carácter general).
- Estructuradas (sesiones de carácter específico).

Tras la toma de contacto, las *entrevistas no estructuradas* (no se esperan respuestas cerradas a las preguntas que se plantean) sirven para familiarizarse con el dominio concreto. Hay que saber conducirlas para evitar divagaciones y anécdotas del experto. Muchos de los datos que se adquieran aquí no serán directamente trasladables al SBC, pero nos ayudarán a hacernos una idea sobre posibles estructuras inferenciales, etc.

Se hará por último una *entrevista estructurada* para cada parte del sistema identificado, donde sí ya se necesitan respuestas concretas. Suelen tener un guión con puntos similares a:

1. Técnicas de análisis basadas en tareas familiares.
   - *a*) Observación directa.
   - *b*) Resolución de casos destacados o difíciles.
2. Técnicas basadas en entrevistas.
   - *a*) No estructuradas.
   - *b*) Estructuradas.
   - *c*) Análisis de casos históricos destacados y difíciles.
3. Técnicas de análisis basadas en tareas especiales.
   - *a*) Técnicas psicológicas para estudiar resolución de problemas.
     1) Análisis de protocolos.
     2) Análisis de toma de decisiones.
     3) Brainstorming.
   - *b*) Técnicas psicológicas para estudiar aprendizaje y memoria.
     1) Resolución de tareas con información limitada.
     2) Resolución de tareas con procedimientos limitados.
   - *c*) Técnicas que enjuician características de los conceptos.
     1) Clasificaciones.
     2) Emparrillado.
     3) Escalonadas.
     4) Escalamiento psicológico.

Cuadro 6.3: Clasificación de los Métodos de Adquisición de Conocimiento.

- √ Descripción general de la tarea.
- √ Descripción de variables involucradas/influyentes.
- √ Reglas generales que ligan a las variables:
  - ♮ Plantilla 1: ¿Por qué? Convierte afirmaciones en reglas.
  - ♮ Plantilla 2: ¿Cómo? Genera reglas de menor nivel.
  - ♮ Plantilla 3: ¿Cuándo? Extrae generalidad de las reglas, generando otras.
  - ♮ Plantilla 4: ¿Qué alternativas hay? Extrae generalidad de las reglas, generando otras.
  - ♮ Plantilla 5: ¿Qué pasaría si? Genera más reglas, con condiciones diferentes.
  - ♮ Plantilla 6: ¿Algo más? Genera más reglas auxiliares o no contempladas.

Es clave tener en cuenta los aspectos no verbales del experto y distinguir sus opiniones personales (sobre todo acerca de la propia ingeniería de conocimiento e inteligencia artificial) de la información objetiva.

Recomendaciones

Las entrevistas deben ser periódicas, hacerse a las mismas horas/días, preferiblemente por la mañana temprano. Es recomendable usar el lugar de trabajo del experto como lugar de reunión, sobre todo al principio, tanto por comodidad como porque tenga sus materiales a mano para consultar.

Resulta muy conveniente ir enseñando al experto los progresos que se van haciendo. Por ello, hay que "digerir" el resultado de una entrevista (realizando un informe para repasarlo con él en la siguiente sesión) antes de concertar otra.

6.2.1. Entrevistas múltiples

Un ingeniero de conocimiento y múltiples expertos

Se da cuando los expertos trabajan en grupo o hay varios tipos de expertos. Si éstos son compatibles, esta clase de entrevistas puede proporcionar muy buenos resultados, por el contraste de opiniones y puntos de vista, lo que asegura un mejor refinamiento y más y mejor conocimiento, de más alto nivel, aunque si la adquisición de conocimiento es de carácter general (no estructuradas), no resulta demasiado útil hacer esto con muchos expertos.

El problema se presenta si los expertos son incompatibles entre sí; en ese caso se debe prescindir siempre de los conflictos y dividir las entrevistas. El problema puede presentarse también si es el ingeniero de conocimiento el que no es capaz de controlar las sesiones (los expertos se centran en un tema, discuten de sus cosas,...) o no se puede concentrar (es muy pesado).

Múltiples ingenieros de conocimiento y un experto

Nunca debería hacerse una entrevista con un solo experto y más de tres ingenieros de conocimiento (no apabullar). Este tipo de entrevistas resulta útil cuando en el equipo de trabajo contamos con ingenieros senior/junior.

La ventaja que tiene la participación de varios ingenieros es que se pueden evitar sesgos introducidos por ellos mismos, cada uno puede aportar cosas, y no se producen vacíos entre la adquisición y la implementación del conocimiento.

La desventaja suele ser que el experto es reticente, pues estas sesiones son más pesadas para él, aunque siempre pueden hacerse más cortas. Lo que como sea debe evitarse son las discusiones entre los propios ingenieros (¡mala imagen!).

Múltiples ingenieros de conocimiento y múltiples expertos

El principal inconveniente de estas entrevistas es que consumen muchísimo tiempo, aunque también aglutina las ventajas de las vistas anteriormente.

Es necesario nombrar un moderador que controle la sesión. Es una de las opciones más usadas.

En general, la técnica de entrevistas para adquirir conocimiento es cara en tiempo, en cualquiera de sus variantes. Es una técnica **introspectiva** (el experto tiene que pensar en lo que sabe y verbalizarlo) y requiere un esfuerzo adicional por parte del ingeniero de conocimiento (que debe hacerse con el vocabulario, confeccionar informes, etc).

Visto por el lado bueno, es una técnica muy general, como hemos dicho, que se puede utilizar en muchos campos y en distintas etapas del desarrollo, sin requerir para ello un entrenamiento especial.

Es clave tener en cuenta los aspectos no verbales del experto y distinguir sus opiniones personales (sobre todo acerca de la propia ingeniería de conocimiento e inteligencia artificial) de la información objetiva.

6.3. El análisis de protocolos

El **análisis de protocolos** es otra técnica de adquisición del conocimiento que requiere algo más de conocimiento por parte del ingeniero de conocimiento, pero tampoco un nivel excesivo.

Consiste en grabar, bien sólo audio o combinado con vídeo, al experto mientras resuelve una tarea, un problema concreto del dominio, motivo por el cual no goza de demasiada aceptación. Sin embargo, al poder acceder a la grabación cualquier ingeniero de conocimiento, se obvian algunos otros problemas de las entrevistas.

Debe quedarle claro al experto que no tiene que explicar lo que va haciendo, sino simplemente verbalizar los pasos que está siguiendo. Pese a ello, el método sigue siendo **introspectivo**.

Es recomendable usar siempre más de una técnica de adquisición del conocimiento, y tanto las entrevistas como el análisis de protocolos son combinables con cualquier otra.

El análisis de protocolos pasa por cuatro fases:

1. *Obtención del protocolo.*- Se dan las instrucciones al experto: verbalizar lo que dice en su cabeza durante la resolución del caso, sin intentar explicarlo. Se pueden tomar notas.
2. *Transcripción y segmentación.*- Escuchar/ver y transcribir lo grabado, separándolo en frases que tengan sentido (desde el punto de vista del conocimiento). No todo el mundo hace las segmentaciones igual, ni una misma persona segmentaría igual una transcripción la primera que sucesivas veces.
3. *Codificación.*- Se identifican objetos, valores, operadores y relaciones. Se obtienen las reglas explícitas en el texto (reglas inferenciales sencillas).
4. *Interpretación.*- Se obtienen las reglas implícitas (cosas que no nos dice directamente), se puede analizar la forma de razonar (progresivo, regresivo,...).

El problema es que un protocolo solo no nos sirve, hay que probar muchos casos. Por esto y por su inherente dificultad, suele usarse exclusivamente en casos puntuales.

Existen variantes:

Análisis del Recuerdo
: Si el experto no es capaz de hablarnos mientras resuelve el problema, puede permitírsele verbalizar el proceso al finalizarlo.

Análisis de las dos Fases
: Consiste en comparar resultados con el propio experto en la fase de codificación.

Análisis del Registro
: Añade una entrevista al final del proceso.

Análisis de Casos Difíciles
: En vez de cualquier caso, se resuelven casos concretos de dificultad específica.

Ventajas

- El flujo de información es unidireccional (en entrevistas era bidireccional), del experto al ingeniero de conocimiento, por lo que se minimizan las interacciones.
- El protocolo puede ser analizado por tantos ingenieros de conocimiento como sea necesario.

- Permite adquirir conocimiento que es difícil de adquirir mediante entrevistas (aspectos relacionados con la estrategia de razonamiento que usa el experto, algo inconsciente que aquí queda reflejado).
- El experto es el reactivo limitante.

Inconvenientes

- Se pueden introducir sesgos (igual que en las entrevistas) inconscientes.
- Es una técnica muy costosa en tiempo (el experto debe aprender a hacer el análisis de protocolos —que no razone sino que actúe—), y es muy larga y pesada para el ingeniero de conocimiento (sobre todo si no tiene experiencia), que debe realizar la transcripción, segmentación, etc.
- Es una técnica introspectiva.

6.4. Cuestionarios

Los **cuestionarios** son una técnica más o menos sencilla que consiste en presentar al experto una serie de fichas u hojas con preguntas concretas (de ahí su utilidad).

Ventajas

- Flujo unidireccional.
- El experto puede contestar el cuestionario cuando desee y donde decida (suelen aceptarla con agrado gracias a esto), por lo que no hay problemas referentes a concierto de reuniones, horarios, etc.
- Consume menos tiempo que las otras técnicas vistas hasta ahora.
- Es útil para describir objetos, jerarquías, certidumbres, relaciones del dominio,...

Inconvenientes

- Su elaboración no es sencilla.
- Es una técnica introspectiva.

6.5. Análisis del movimiento de ojos

El **análisis del movimientode ojos** es una técnica de adquisición del conocimiento que sólo se puede usar en los campos en que sea necesario un reconocimiento visual del problema por parte del experto (existe un aparato —*Eye Mark Recorder*— que registra la dirección, posición y duración de la mirada).

Los datos que se obtienen no tienen por qué ser directamente trasladables al sistema/dominio, aunque sí pueden dar información sobre el comportamiento del experto, siendo así un buen punto de partida. Puede ser clave para descubrir la diferencia entre lo que se hace y lo que hay que hacer (que suele ser lo que el experto comunica).

Es una técnica **no introspectiva**, cuyo principal inconveniente es su carestía.

6.6. Método de observación directa

Esta técnica de adquisición del conocimiento es siempre recomendable. La **observación directa** consiste en acudir al lugar de trabajo del experto y observar allí su comportamiento.

Su interés reside en que se ve lo que realmente pasa, eliminando las interpretaciones subjetivas sobre el trabajo del experto. Es un método **no introspectivo**, aunque siempre se pueden hacer preguntas y, por supuesto, tomar notas. Puede ser interesante concertar una entrevista después para aclarar dudas.

Su objetivo fundamental es captar conocimiento procesal, siendo útil para refinar el sistema (no en conocimiento, pero sí a lo mejor en aspectos relacionados con la interfaz, por ejemplo). Su principal inconveniente, la gran cantidad de tiempo del ingeniero de conocimiento que consume, aunque no afecta en absoluto al experto.

6.7. Extracción de curvas cerradas

Esta técnica suele usarse en campos en los que el conocimiento visual es importante, siendo especialmente interesante cuando existen relaciones espaciales entre los elementos del dominio, que se desean extraer.

Consiste en confeccionar cartulinas con representaciones de los objetos del dominio (hasta un máximo, por ejemplo, 50) y pedir al experto que relacione, rodeándolos, encerrándolos en un círculo, aquéllos que según su criterio formen patrones, aparezcan ligados, tengan características similares, etc. Esta fase puede repetirse cuantas veces se desee, aplicando distintos criterios.

Es obvio que el conocimiento obtenido por medio de la **extracción de curvas cerradas** puede no ser directamente trasladable al sistema experto, pero sin duda, ayuda en el proceso de establecimiento de relaciones entre los conceptos del dominio y/o para obtener clasificaciones.

Es una técnica bastante fácil de utilizar para el ingeniero de conocimiento, aunque requiere conocer con relativa profundidad el campo en que trabajamos y puede aparecer el problema de que el criterio de clasificación sea difícil de explicitar por parte del experto.

6.8. Las técnicas de escalamiento psicológico

Las **técnicas de escalamiento psicológico** son una serie de métodos semiautomáticos de adquisición de conocimiento que constituyen el conjunto de los más usados

en IC. Todas ellas tienen el mismo formato de datos a la entrada, una *matriz triangular superior*:

	E_1	E_2	E_3	$\ldots$	E_n
E_1	0	d_{12}	d_{13}	$\ldots$	d_{1n}
E_2		0	d_{23}	$\ldots$	d_{2n}
$\vdots$			$\ddots$		$\vdots$
E_{n-1}				0	$d_{(n-1)n}$
E_n					0

donde d_{ij} son **juicios de distancia** que representan cómo se parecen/diferencian los elementos i y j del dominio.

Las salidas de los tres métodos de escalamiento psicológico que veremos son distintas, pero siempre están igual formateadas (la misma entrada y datos producen siempre la misma salida), y hay una relación constante entre ellas.

No obstante, la aplicación de estas técnicas no es sencilla, pues la parte manual, encargada de adquirir los elementos del dominio (del experto, mediante entrevistas) es muy importante: el conjunto de elementos debe ser completo y consistente, o de lo contrario el conocimiento que se obtenga será incompleto, incorrecto y/o inconsistente, ya que las distancias están condicionadas por el contexto de propios elementos. En el otro extremo, elegir demasiados se enfrentará con la negativa y/o imposibilidad del experto de calcular todas las distancias.

Una vez que se tienen los elementos del dominio, hay que obtener las $\frac{n(n-1)}{2}$ distancias (si n es el número de elementos a manejar). Para ello existen técnicas auxiliares para suplir el hecho de tener que mostrar al experto la tabla directamente:

- √ Clasificaciones: consisten en dibujar fichas que representen los elementos del dominio y pedir al experto que haga grupos disjuntos con ellos, repitiendo el proceso en varias ocasiones utilizando distintos criterios. La distancia entre dos elementos será entonces el inverso del número de veces que esos dos elementos se clasificaron juntos.
- √ Emparrillado: es un método matemático semiautomático para calcular las distancias que veremos en más profundidad.

6.8.1. Escalamiento multidimensional (*EDM*)

Existen programas estadísticos que contienen herramientas para llevar a cabo un escalamiento multidimensional sobre un conjunto de datos. El **EDM** intenta colocar los elementos de la matriz en un espacio de la menor dimensionalidad posible, reduciendo al máximo la dimensionalidad de la matriz.

El problema que tiene es que a veces no es fácil interpretar los resultados, y si la salida tiene más de 3 dimensiones no se puede representar.

	Coruna	*Lugo*	*Santiago*	*Orense*	*Pontevedra*	*Vigo*
Coruna	0	90	60	170	130	150
Lugo		0	120	110	190	210
Santiago			0	120	70	90
Orense				0	100	100
Pontevedra					0	20
Vigo						0

	Latitud	*Longitud*
Coruna	75	−20
Lugo	40	60
Santiago	20	−20
Orense	−60	80
Pontevedra	−55	−20
Vigo	−75	−20

Cuadro 6.4: Ejemplo de aplicación de EDM.

Además, existen infinitas salidas, porque el método sólo fija el origen de coordenadas y no la ubicación de los elementos (hay, por tanto, infinitas orientaciones, infinitas rotaciones de los ejes), lo que genera la dificultad en la interpretación.

No obstante, ayuda a descompilar conocimiento de los expertos, aunque también puede no ser directamente trasladable al sistema experto.

6.8.2. Análisis de clusters (*Clustering*)

El **clustering** consiste en agrupar los elementos que están más cerca entre sí. En este caso, se buscan los dos elementos más próximos, se agrupan formando un nuevo elemento, se eliminan de filas y columnas, se añade el nuevo elemento a la matriz y se recalculan las distancias con respecto al resto de elementos, repitiendo el proceso hasta que sólo queden dos elementos en la matriz, o bien hasta que en número de *clases* generadas sea adecuado en nuestro dominio. El resultado final será un **dendrograma** o *árbol jerárquico* (ver figura 6.5).

La cuestión más delicada en este caso es la forma de recalcular las distancias: ¿cómo lo hacemos? ¿escogiendo el máximo? ¿el mínimo? ¿una media? La decisión dependerá de lo que sea más conveniente en el contexto en que estemos trabajando. Esta técnica también se incorpora en muchos paquetes estadísticos. Además, como se puede observar, una vez obtenida la matriz de elementos con sus distancias, ninguna de las técnicas de escalamiento es introspectiva.

	E_1	E_2	E_3	E_4	E_5	E_6	E_7
E_1		10	10	7	6	13	8
E_2			4	7	8	**2**	8
E_3				9	12	3	8
E_4					5	5	5
E_5						6	6
E_6							9
E_7							

	E_1	E_3	E_4	E_5	E_7	(E_2, E_6)
E_1		10	7	6	8	10
E_3			9	12	8	**3**
E_4				5	5	5
E_5					6	6
E_7						8
(E_2, E_6)						

	E_1	E_4	E_5	E_7	$((E_2, E_6), E_3)$
E_1		7	6	8	10
E_4			**5**	5	5
E_5				6	6
E_7					8
$((E_2, E_6), E_3)$					

	E_1	(E_4, E_5)	E_7	$((E_2, E_6), E_3)$
E_1		6	8	10
(E_4, E_5)			**5**	5
E_7				8
$((E_2, E_6), E_3)$				

	E_1	$((E_4, E_5), E_7)$	$((E_2, E_6), E_3)$
E_1		6	10
$((E_4, E_5), E_7)$			**5**
$((E_2, E_6), E_3)$			

	E_1	$(((E_4, E_5), E_7), ((E_2, E_6), E_3))$
E_1		6
$(((E_4, E_5), E_7), ((E_2, E_6), E_3))$		

Cuadro 6.5: Ejemplo de clustering (I).

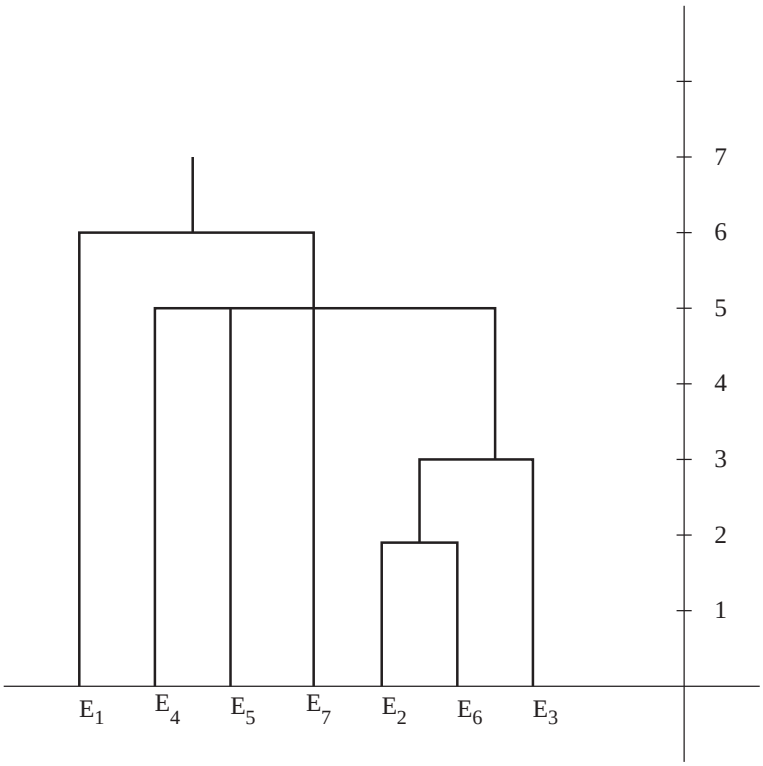


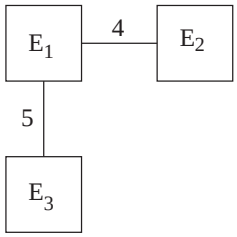
Figura 6.5: Ejemplo de clustering (y II): dendrograma.

El *clustering* permite elicitar muy rápido conocimiento muy jerarquizado en la mente del experto. Es útil también para validación, para construcción de interfaces hombre-máquina (comparación de opiniones sobre la importancia de diferentes aspectos), para comprobar el nivel de nuestro sistema frente a los expertos, e incluso para verificar si las respuestas de un grupo de expertos están al mismo nivel.

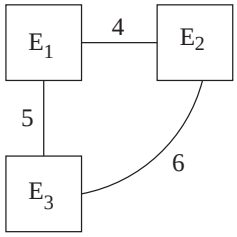
6.8.3. Redes ponderadas (*Pathfinder*)

Esta última es la menos usada de las tres técnicas de escalamiento psicológico que vamos a ver. Su salida, como su propio nombre indica, es una **red ponderada** en la que los nodos son los elementos del dominio y los arcos que los unen aparecen ponderados por un peso que es la distancia que los separa. Sólo existirá un arco entre dos nodos si la distancia entre ellos a través de cualquier camino es mayor que el valor especificado entre ambos en la matriz de entrada:

	E_1	E_2	E_3
E_1		4	5
E_2			13
E_3			



	E_1	E_2	E_3
E_1		4	5
E_2			6
E_3			



Cuadro 6.6: Ejemplo de redes ponderadas.

El *pathfinder* es muy útil si la representación del conocimiento en nuestro sistema utiliza algún tipo de red (por ejemplo, una red semántica), ya que la traducción de conocimiento del dominio es entonces mucho más directa.

En general, las **ventajas** del escalamiento psicológico son que proporciona contenido y a veces incluso arquitectura a nuestra base de conocimientos. Permiten comparar conocimientos, opiniones y criterios de varios expertos sobre un mismo conjunto de elementos, además de proporcionar un método riguroso para combinar conocimiento que procede de distintos expertos o incluso del cliente. No son técnicas demasiado introspectivas (salvando la etapa de determinación de elementos y distancias entre ellos), y están libres de cualquier interpretación que pudiese ser incluida por parte del ingeniero de conocimiento. La salida es estándar y tiene un formato riguroso, lo que hace que diferentes ingenieros de conocimiento con distintos expertos deban obtener los mismos resultados, algo que es un buen paso hacia la reutilización (además de permitir la automatización). Por último, son útiles no sólo en adquisición sino también en validación y construcción de interfaces de usuario.

En cuanto a los **inconvenientes**, cada técnica tiene los suyos. En general, los elementos y las distancias poseen un proceso de obtención no trivial. Como hemos visto, los *EDM* no son fáciles de interpretar y si su salida es gráfica no es sencilla de etiquetar. En *clustering* puede no ser directo decidir qué criterio (máximo, mínimo, media) escoger para el recálculo de distancias, al igual que en *redes semánticas*.

Si se usa una herramienta, debemos familiarizarnos con ella; hay que tener una base matemático-estadística para poder saber si es correcto lo que hacemos y saber interpretarlo. Por último, si no existe una estructura de ligazón fuerte entre elementos, o no hay estas relaciones, estas técnicas son inútiles.

6.9. La teoría de constructos personalizados: el Emparrillado

La siguiente técnica que veremos, denominada **técnica de constructos personalizados** o **emparrillado** (**repertory grid**), no es propiamente una técnica de adquisición de conocimiento, sino más bien un método auxiliar de clasificación. En la práctica se usan herramientas que la implementan, que interrogan al usuario sobre los elementos del dominio, sus relaciones, etc. con el fin de estructurar el conocimiento del experto de una determinada manera (es pues, un método semiautomático). Nosotros estudiaremos su funcionamiento interno.

El *emparrillado* fue desarrollado inicialmente por George Kelly, un psicólogo clínico que defendía que cada persona organiza sus conocimientos de una manera distinta y además cambiante con el tiempo, y reflejaba las opiniones de las personas sobre las cosas en forma de **constructos personalizados**. Su modelo cognitivo utilizado para razonar con dichos constructos sería más tarde aplicado por Shaw y Griness a la IC (SS.EE. *Planet*, 1982), siendo así una de las primeras técnicas de aproximación automática para tratar el problema de la adquisición del conocimiento.

Al estar basada en un modelo del pensamiento humano, esta es una técnica muy potente que sirve para tener estructurado y accesible el conocimiento de una persona. Incluye un diálogo inicial con el experto, una sesión de valoración y un análisis de los resultados (es decir, de los grupos, los conceptos y las dimensiones).

Se utilizan dos vertientes: *técnicas binarias*, que permiten clasificación simple, y *multivaluadas*, que proporcionan una clasificación más rica, al ser continua, dando opción a la inclusión de nociones de incertidumbre e incluso conjuntos difusos.

Este método posibilita la obtención de información que el experto no es consciente que conoce (no es introspectiva, por tanto, salvo en su etapa inicial, aunque puede considerarse que es intrusiva, pues desvela al experto eso que no es consciente que sabía). Por otra parte, matemáticamente, la parrilla es una aplicación de los elementos del dominio en un conjunto de características. Como podemos intuir, la primera fase, de obtención de elementos y características (constructos: $[c_i, \bar{c}_i]$) es el momento más peliagudo, aunque posteriormente existen fórmulas y ecuaciones que ayudan a eliminar redundancias e informaciones poco significativas (juicios de umbrales, etc).

Una *parrilla* es una matriz donde las columnas son los elementos del dominio y las filas representan los constructos, que son características en las que deseamos clasificar. Dichos constructos son bipolares, representando una oposición de conceptos, sirviéndonos la matriz para ver cómo piensa el experto, cómo se organiza.

	E_i		
c_j	v_{ij}	...	$\bar{c}_j$

donde

E_i	son los elementos del dominio identificados por el experto
c_j, $\bar{c}_j$	son, respectivamente, características y su contrapartida en el dominio (su opuesto lógico)

y

v_{ij}	son los valores que se corresponden por la relación entre unos y otros.

Cuadro 6.7: Esquema de una parrilla.

El emparrillado consta de cinco fases diferenciadas:

1. Identificación de elementos (E_i).
2. Identificación de características (c_j).
3. Diseño de la parrilla.
4. Formalización de la parrilla.
5. Interpretación de resultados.

6.9.1. Identificación de elementos E_i

Como ya vimos en escalamiento psicológico, los elementos del dominio se obtienen mediante entrevista con el experto y debe lograrse un conjunto completo, consistente y representativo. Las sesiones pueden ser más o menos hábiles, pero los elementos

obtenidos deben ser homogéneos, representativos y tener en cuenta que no es útil manejar parrillas demasiado amplias, aunque se pueden manejar varias, lo que también plantea la cuestión de cómo "segmentar" los elementos (normalmente, se agruparán en una misma parrilla los más homogéneos a su vez entre sí).

6.9.2. Identificación de características c_j

Las características se obtienen de la misma manera que los elementos. Deben poder expresarse como un par de conceptos antagónicos, con el fin de poder manejar un segmento clasificatorio. No tienen por qué ser uno el polo negativo del otro, pero sí su negación lógica en el contexto del dominio. Si el experto no fuese luego capaz de clasificar utilizando los conceptos establecidos sería indicativo de que éstos están mal elegidos, no son los que él usa, le estaríamos obligando a clasificar según otros cánones, de manera que habría que replanteárselos.

Para llevar a cabo la selección de características se usa normalmente el método de las **tríadas** , que consiste en tomar los elementos del dominio de tres en tres y presentarlos al experto, con la propuesta de que enuncie una característica que diferencia a dos de ellos del tercero. Se ha estudiado que cognitivamente este sistema de elección es mucho más eficaz, pues el hecho de que se muestren tres elementos evita sesgos que se producirían con dos y ayuda a elaborar los constructos por oposición y no por negación.

No obstante, también se puede utilizar *extracción de curvas cerradas*, que ya vimos, e incluso simplemente *entrevistas*. Siempre es mejor la técnica de las tríadas, pero se puede simultarear con alguna de éstas (o con ambas) para contrastar, pues siempre alguna puede funcionar mejor que otra con según qué expertos.

6.9.3. Diseño de la parrilla

Una vez obtenidos elementos del dominio y características (*constructos*), hay que dar valor a la parrilla. Hay varias formas de construirla, que citamos a continuación de menos a más usada:

Dicotómica
: Se clasifica cada E_i dependiendo de si "tiene" o no c_j (asignación binaria).

Clasificatoria
: Se clasifican los n E_i en un intervalo $1..n$ de forma que $v_{ij} \in [1, n]$ representa el orden del elemento con respecto al constructo (su proximidad al constructo). Se obtiene de esta manera una escala de clasificación o vector clasificador.

Evaluativa
: Se elige una escala que se adapte a los elementos y se les clasifica según ella, pudiendo repetirse valores. Representa cómo ve el experto al elemento dentro de ese constructo, no siendo en este caso una escalación de elementos.

Estos distintos métodos de diseño de la parrilla no son excluyentes, de hecho, los resultados deberían ser compatibles independientemente de qué forma de construcción usásemos. En general, siempre es preferible una aproximación por rangos a una binaria, ya que hay técnicas de aplicabilidad posterior que no son compatibles con datos bipolares.

6.9.4. Formalización

Una vez diseñada la parrilla, tenemos una matriz en la que las filas nos muestran cómo varía una característica de un elemento a otro, y las columnas cómo se comportan los elementos en el dominio (según las características seleccionadas). Es decir, la tabla indica la participación de un elemento en una característica: por columnas, la participación de un elemento en todas las características, y por filas, la participación de todos los elementos en una determinada característica (constructo):

	E_1	$\ldots$	E_i
c_1	v_{11}	$\ldots$	v_{i1}
$\vdots$	$\vdots$	$\ddots$	
c_j	v_{1j}	$\ldots$	v_{ij}

La parrilla ha de explorarse en dos sentidos, horizontal y vertical, para obtener dos árboles jerárquicos, uno de elementos y otro de características, y ver cómo están relacionados. Esta fase puede automatizarse completamente.

Clasificación de elementos

Analizando la matriz por columnas (elementos), calculamos distancias entre elementos para obtener un árbol:

	E_1	E_2	E_3	$\ldots$	E_8
c_1	2	3			
c_2	2	2			
c_3	4	5			
c_4	5	4			
c_5	5	5		$\ldots$	
c_6	4	4			
c_7	2	2			
c_8	4	2			
c_9	1	1			

asumiendo un rango [1.,5], hay varias formas de calcular las distancias, pero la que se suele usar es la suma de diferencias en valor absoluto de los valores de los elementos para cada característica:

$$d(E_1, E_2) = d(E_2, E_1) = 1 + 0 + 1 + 1 + 0 + 0 + 0 + 2 + 0 = 5$$

De esta manera se obtiene una matriz triangular superior de distancias entre elementos, y para construir el árbol (*dendrograma*) se siguen los pasos que ya vimos en escalamiento psicológico: se buscan los dos elementos más próximos, se eliminan de la tabla, se agrupan, se recalculan distancias,...).

Clasificación de características

Para las características también se va a obtener un árbol clasificatorio, pero en esta ocasión, como manejamos dos polos, tendremos que calcular dos distancias:

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	
Manejable	2	3	5	2	5	3	3	2	No manejable
Deportivo	2	2	2	1	5	2	3	1	No deportivo
$\bar{Deportivo}$	4	4	4	5	1	4	3	5	$\bar{Nodeportivo}$
$\vdots$				$\ldots$					

La regla de cálculo es la misma:

$$d_1(c_i, c_j) = 0 + 1 + 3 + 1 + 0 + 1 + 0 + 1 = 7$$

$$d_2(c_i, c_j) = d_1(c_i, \bar{c_j})$$

$$d_2(c_i, c_j) = 2 + 1 + 1 + 3 + 4 + 1 + 0 + 3 = 15$$

donde $\bar{c_j}$ se calcula a partir de c_j. Si se aplicó una construcción dicotómica al elaborar la parrilla, simplemente se cambian los "polos" (se otorgan los valores complementarios); si fue clasificatoria, se invierte el orden, y si fue evaluativa, se sigue misma dinámica que en el siguiente ejemplo:

c_j	1	2	3	4	5
$\bar{c_j}$	5	4	3	2	1

Una vez calculadas las dos distancias para cada par de características por elemento, tendremos una matriz construida de la siguiente forma:

	E_1	E_2	$\ldots$	E_n	
c_1	$\ddots$		$d_1(c_i, c_j)$		$\bar{c_1}$
c_2					$\bar{c_2}$
$\vdots$	$d_1(c_i, \bar{c_j}) = d_2(c_i, c_j)$		$\ddots$		
c_i				$\ddots$	$\bar{c_i}$

donde la submatriz superior son las distancias "directas" (d_1) y la submatriz inferior, las distancias "inversas" (d_2).

El paso a una matriz triangular superior se consigue eligiendo la distancia mínima de las dos (d_1, d_2), y a partir de ahí para obtener el dendrograma el proceso es el que ya conocemos.

6.9.5. Análisis y estudio de los resultados obtenidos

Fundamentalmente, debemos analizar los árboles de elementos y características. Estudiando el primero de ellos, los problemas posibles que podemos encontrar son:

- √ *Dos elementos están muy juntos* en el árbol y el experto afirma que no deberían.
  - ↪ Debemos comprobar que los valores de la parrilla son correctos y que los cálculos están bien hechos.
  - ↪ Falta una característica diferenciadora en la parrilla.
- √ *Dos elementos están muy separados* en el árbol y el experto afirma que no deberían.
  - ↪ Debemos comprobar que los valores de la parrilla son correctos y que los cálculos están bien hechos.
  - ↪ Falta una característica conciliadora en la parrilla.
- √ *Dos características (constructos) aparecen muy ligadas* y no deberían, según el experto.
  - ↪ Debemos comprobar que los valores de la parrilla son correctos y que los cálculos están bien hechos.
  - ↪ Falta un elemento diferenciador (que participa de una y no de la otra) en la parrilla.

Por su parte, en el árbol de características se estudian también las características por pares, empezando por las más próximas, buscando relaciones entre ellas. El objetivo es afinar el polo.

Paralelas

Son características (A, B) y (X, Y) que se comportan:

$$\begin{array}{rcl} A & \rightarrow & X \\ B & \rightarrow & Y \end{array}$$

como por ejemplo $(Familiar, Coupe)$ y $(Habitable, NoHabitable)$, ya que

$$\begin{array}{rcl} Familiar & \rightarrow & Habitable \\ Coupe & \rightarrow & NoHabitable \end{array}$$

sin embargo

$$\begin{array}{lcl} Habitable & \nrightarrow & Familiar \\ NoHabitable & \nrightarrow & Coupe \end{array}$$

En estos casos, de acuerdo con el experto, se pueden resumir ambas características en una que mantenga este comportamiento.

Recíprocas

Son características (A, B) y (X, Y) que se comportan:

$$\begin{array}{lcl} A & \leftrightarrow & X \\ B & \leftrightarrow & Y \end{array}$$

como se da por ejemplo en el caso de $(GranCilindrada, PocaCilindrada)$ y $(Potente, PocoPotente)$, pues

$$\begin{array}{lcl} GranCilindrada & \leftrightarrow & Potente \\ PocaCilindrada & \leftrightarrow & PocoPotente \end{array}$$

En estos casos las características son típicamente redundantes, puede eliminarse una de ellas o resumir ambas en una nueva.

Ortogonales

Son características (A, B) y (X, Y) que se comportan:

$$\begin{array}{lcl} A & \rightarrow & X \\ B & \nrightarrow & Y \end{array} \quad o \quad \begin{array}{lcl} A & \rightarrow & X \\ A & \nrightarrow & Y \\ & & \\ B & \rightarrow & X \\ B & \nrightarrow & Y \end{array}$$

como por ejemplo $(Deportivo, NoDeportivo)$ y $(Nervioso, Pesado)$, ya que

$$\begin{array}{lcl} Deportivo & \rightarrow & Nervioso \\ Deportivo & \nrightarrow & Pesado \\ NoDeportivo & \nrightarrow & Nervioso \\ NoDeportivo & \nrightarrow & Pesado \end{array}$$

En este caso o bien uno de los polos implica relación, o bien ambos implican una, el caso es que algún polo queda "suelto" (permite eliminar uno de los polos).

Ambiguas

Son características (A, B) y (X, Y) que se comportan:

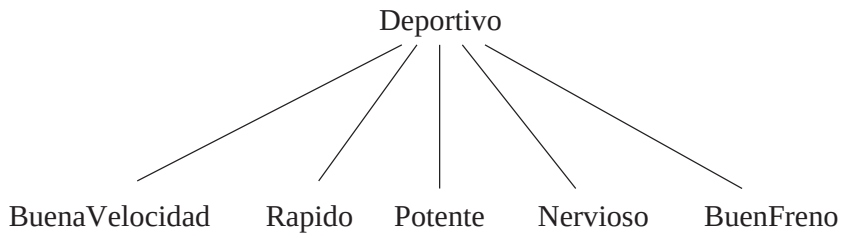
$$\begin{array}{lcl} A & \rightarrow & X \\ B & \rightarrow & X \\ B & \rightarrow & Y \end{array} \quad o \quad \begin{array}{lcl} A & \rightarrow & X \\ A & \rightarrow & Y \\ B & \rightarrow & X \\ B & \rightarrow & Y \end{array}$$

como por ejemplo $(Seguro, Ligero)$ y $(Deportivo, NoDeportivo)$, ya que

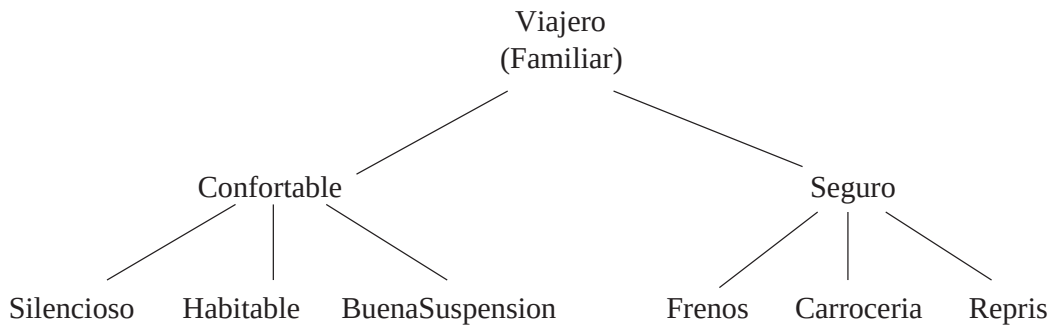
$$\begin{array}{lcl} Deportivo & \rightarrow & Seguro \\ Deportivo & \rightarrow & Ligero \\ NoDeportivo & \rightarrow & Seguro \\ NoDeporitvo & \rightarrow & Ligero \end{array}$$

No queda muy clara la relación entre las características (alguno de los polos no está bien definido), se puede estudiar si es posible cambiar alguna de ellas.

También se suelen construir árboles de *por qué* y *cómo*, que sirven para obtener subconjuntos y superconjuntos de características.



(a) Ejemplo de árbol *por qué*.



(b) Ejemplo de árbol *cómo*.

El emparrillado es, por tanto, como ya hemos dicho, útil en clasificación o para catalogar expertos. Ayuda también en ámbitos de incertidumbre, o a la validación.

Sus problemas residen en que los datos aportados por el experto son subjetivos, personales (surgirán distintas parrillas con diferentes expertos; el grado de distanciamiento entre ellos marca su nivel de similitud).

6.10. Técnicas especiales de adquisición de conocimiento en grupo

Para cerrar este capítulo hablaremos de una serie de técnicas especiales a usar cuando queremos trabajar con un grupo de expertos al mismo tiempo.

Sus ventajas e inconvenientes ya fueron mencionados cuando hablamos de las entrevistas múltiples (página 58): el precio de obtener una bases de conocimiento más completas y tener la seguridad de que se han tratado todos los aspectos relevantes del dominio/problema es la introspección y el esfuerzo general por ambas partes.

Las técnicas de adquisición de conocimiento en grupo más destacables son:

1. Tormenta de ideas (*brainstorming*).
2. Técnica nominal de grupo.
3. Método Delphi.
4. Entrevistas.
5. Emparrillado.
6. Escalamiento psicológico.

6.10.1. Tormenta de ideas (*Brainstorming*)

En adquisición del conocimiento, para aplicar esta popular técnica, basada en la opinión sajona de que la cantidad mejora la calidad, intentaremos tener varios tipos de expertos e ingenieros de conocimiento, que expondrán sus ideas (con una breve explicación) sin ningún tipo de crítica. Suele ser favorable la presencia de un moderador y su utilidad es manifiesta en dominios en los que se requiere inventiva.

Una vez recopiladas las ideas, se llevan a cabo una serie de cribas. Las primeras se suelen basar en la aplicabilidad inmediata de la solución propuesta, el ajuste con respecto a restricciones existentes (de tiempo, de coste,...), la compatibilidad con otros aspectos del problema (que no supongan conflictos con otros elementos ya implementados/implantados, etc). Reducida la lista, en último caso se puede elegir la definitiva por votación.

6.10.2. Técnica nominal de grupo

Esta técnica es exactamente igual que la anterior, pero en ella las ideas se exponen por escrito. Es mejor cuando hay gente tímida en el grupo, o que no acepta críticas, aplicable fundamentalmente cuando se trata con poca gente.

6.10.3. Método Delphi

El método Delphi también es un método por escrito, basado en cuestionarios confeccionados por los ingenieros de conocimiento sobre aspectos del problema que se presentan a distintos tipos de expertos.

Los expertos responderán el cuestionario de forma anónima, y también a un cuestionario sobre los cuestionarios (¿son las preguntas relevantes? ¿importantes? ¿sobran? ¿faltan? ¿cuáles?). Esto ayuda a refinarlos, acometiendo una segunda vuelta acompañada de los resultados de la anterior (media y varianza —primer y tercer cuartil—). Se repite iterativamente (aunque con dos vueltas suele ser suficiente) hasta obtener una respuesta más o menos homogénea —acuerdo—. Si existen dispersiones importantes, habría que pasar a algo no anónimo para identificar el problema, aunque no suele ser necesario.

Capítulo 7

Evaluación de los sistemas basados en conocimiento

La **evaluación** de los Sistemas Basados en Conocimiento consta de 4 aspectos que podemos estudiar:

Verificación
: Trata de estudiar la *corrección* formal de nuestro sistema, algo que puede hacerse tanto durante el desarrollo como al final del mismo. De los cuatro aspectos, es el único criterio estático. La mayoría de las herramientas, como por ejemplo Nexpert incluyen corrección de sintaxis.

Validación
: Se trata de ver si el sistema es correcto, si contempla todos los casos, si el modelo computable es *válido*. Para poder realizarse, el sistema debe estar funcionando, por tanto. No es suficiente realizar validación por módulos, es imprescindible una prueba global integrada.

Usabilidad
: *Usabilidad* y *Utilidad* son conceptos relativamente recientes, influencia de la ingeniería del software. El primero representa la satisfacción que tiene el usuario con el sistema, la interfaz, el tipo de respuestas, su redacción, adecuación a su nivel, etc. Es, pues, un criterio dinámico (el sistema debe estar funcionando).

Utilidad
: Por último, la *utilidad* es un criterio también dinámico que intenta analizar el cambio/mejoras que se producen en el entorno/empresa en que se introduce el sistema, ver si satisface las expectativas. Se repasan las propuestas realizadas con el modelo de organización/contextual (motivos que impulsaron al desarrollo del sistema) y se comprueba si se han cumplido.

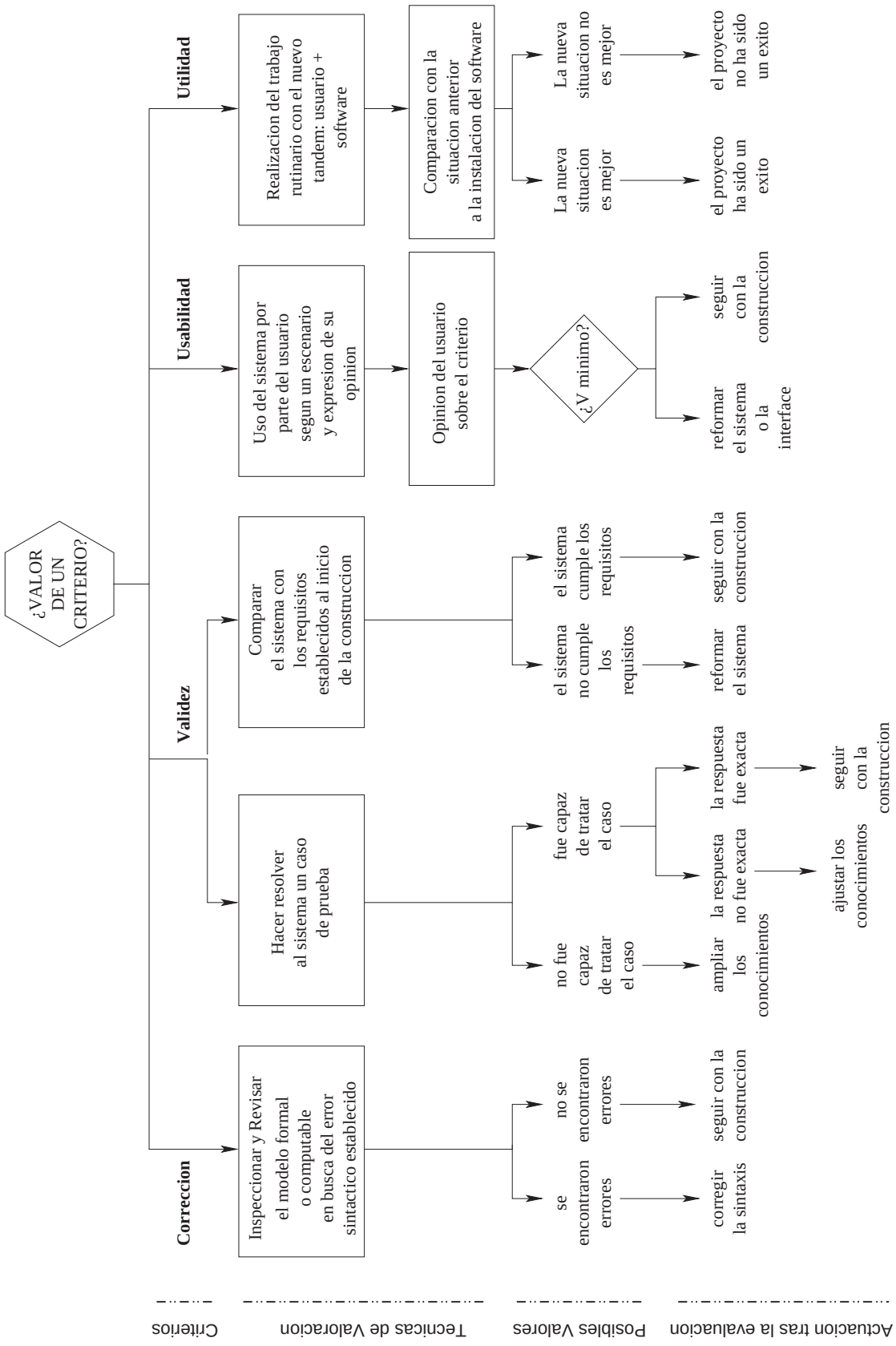
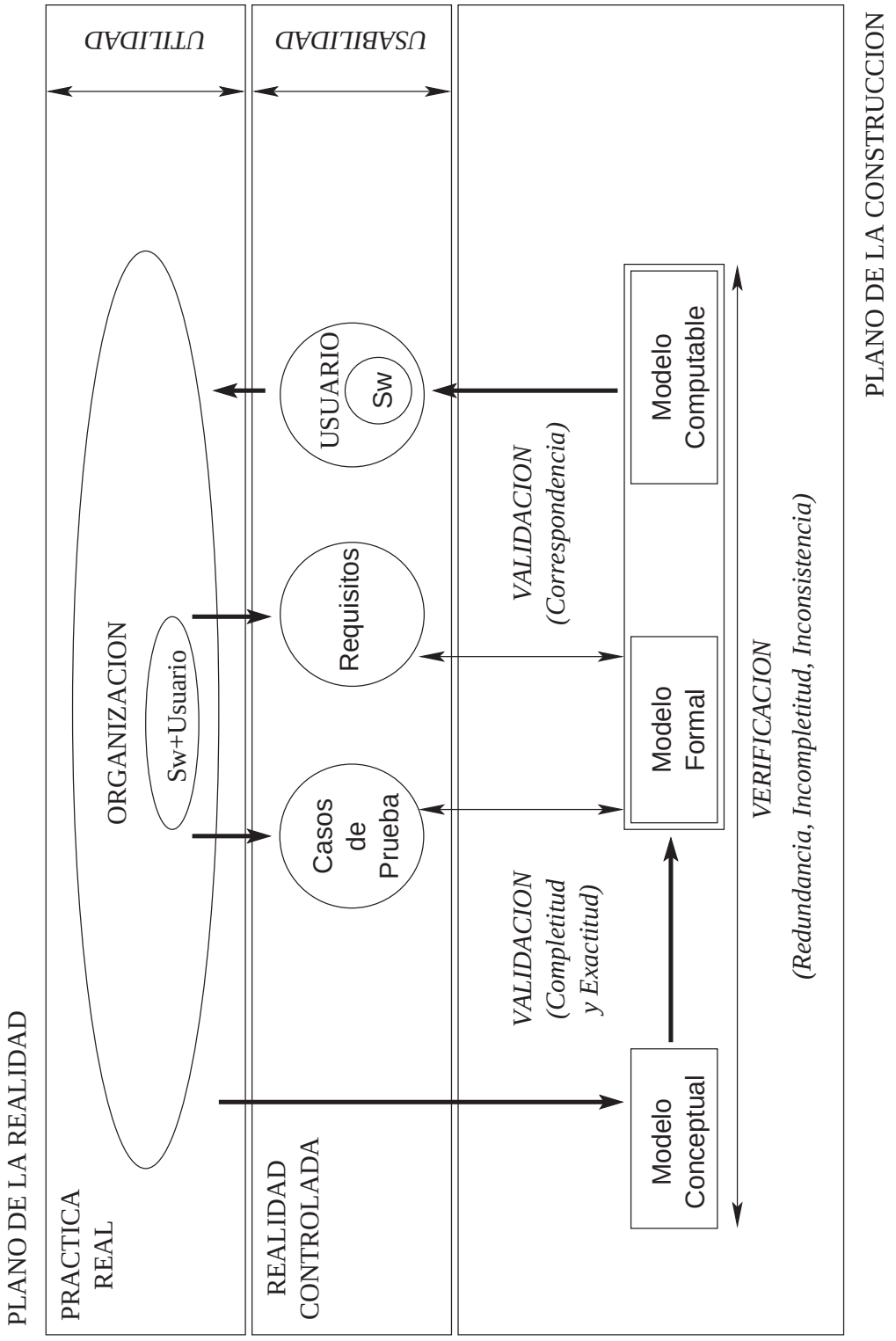


Figura 7.1: Técnicas de valoración para SS.EE.



Verificacion: Aspectos mas internos del sistema.
Validacion: Aspectos del contexto en un entorno "controlado".
Usabilidad: Amplia el entorno al usuario.
Utilidad: Valora el sistema en el mundo exterior.

Figura 7.2: Relación entre los distintos tipos de evaluación.

7.1. Verificación y validación

7.1.1. Sistemas de verificación automática

Todos las pruebas de verificación de software estándares que existen en ingeniería del softwae son aplicables a los SBC y SSEE, aunque son necesarias algunas a mayores. Existen, como hemos mencionado, herramientas que proporcionan verificación simple de estructuras, principalmente sintáctica. La mayoría de los sistemas de verificación automática son para sistemas basados en reglas y suelen controlar:

Inconsistencias.- Pueden aparecer por propio error o al trabajar varios ingenieros sobre una misma base de conocimientos.

$$\begin{aligned} Base\_Conocimientos + Base\_Hechos &\Rightarrow P \wedge \neg P \\ atributo(a,x) \wedge atributo(a,y) &\Rightarrow C \end{aligned}$$

Redundancias.- Pueden ser o no determinantes (puede haber que eliminarlas o ser correctas/necesarias en nuestro sistema), reduciendo sobre todo la eficiencia.

$$\begin{array}{llll} & & & R_i) \quad \alpha \Rightarrow \beta \\ R_i) & \alpha \Rightarrow \beta & & R_j) \quad \beta \Rightarrow \gamma \\ R_j) & \alpha \Rightarrow \beta & & R_k) \quad \gamma \Rightarrow \theta \\ & & & R_r) \quad \alpha \Rightarrow \theta \end{array}$$

Subsunción.- Si las conclusiones de dos reglas son iguales y las premisas de las condiciones son una un subconjunto de la regla se dice que hay *subsunción*. Es decir, una regla está *subsumida* en otra si ambas tienen la misma conclusión y las premisas de una son un subconjunto de las premisas de la otra (que es más general). Igual que en el caso anterior, pueden querer eliminarse o no, dependiendo del caso.

Cadenas circulares.- Son situaciones del tipo:

$$\begin{array}{llll} \alpha & \wedge & \eta_0 & \Rightarrow \beta_1 \\ \eta_0 & \wedge & \beta_1 & \Rightarrow \beta_2 \\ & \cdots & & \\ \eta_{n-1} & \wedge & \beta_n & \Rightarrow \alpha \end{array}$$

Reglas no disparables.- Si las premisas que tiene una regla nunca son una conclusión de otra ni son hechos iniciales, la regla en cuestión nunca se activará. Con frecuencia es un error debido a que se escribió mal esa regla o por falta de conocimiento (más reglas).

Conclusiones no alcanzables.- Si la conclusión de una regla no es parte de un antecedente de otra regla (por ejemplo, en encadenamiento regresivo) nunca se llegará a ella. Igual que en el caso anterior (de hecho, se pueden considerar un tipo especial de reglas no disparables), o bien está mal escrita, o bien falta conocimiento.

Completitud.- Si la base de conocimientos no está completa puede darse la situación en que no hayamos alcanzado ninguna conclusión y tampoco quede ninguna regla por ejecutar.

Hay cuatro tipos principales de sistemas de verificación automática:

- √ *Sistemas de verificación tabular*: se basan en la colocación de las reglas en tablas, agrupándolas por las conclusiones (en una misma fila, la primera columna contiene una conclusión c_i y la segunda, las reglas que concluyen algo sobre c_i), para comprobar posteriormente, dos a dos, la estructura de dichas reglas, pudiendo así detectar redundancias e incluso inconsistencias (si además de c_i tienen en cuenta la negación de c_i).
- √ *Sistemas de propagación de restricciones*: toman un conjunto de hechos iniciales e intermedios lo más completo posible y lo propagan por el sistema a travéz de sus restricciones, comprobando el resultado (analizando si hay redundancias, etc).
- √ *Sistemas de verificación basados en redes de Petri*: combierten la base de conocimientos de reglas en una red de Petri , en la que hipótesis y conclusiones son nodos y las transiciones representan reglas. Estas re-

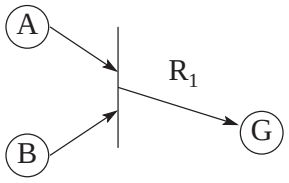


Figura 7.3: Ejemplo de red de Petri.

des sirven para comprobar la accesibilidad de las conclusiones, ya que son expresables en forma de sistemas de ecuaciones lineales. Además, una vez se dispone del sistema de ecuaciones, si se introduce una inconsistencia y el sistema tiene solución, significa que el sistema tiene inconsistencias (está mal); por el contrario, si no la tiene, es que el sistema es correcto.
El peor inconveniente de este tipo de sistemas de verificación es que su complejidad es exponencial, aunque a pesar de ello son de los más empleados.

- √ *Sistemas de verificación basados en grafos dirigidos*: a partir de la base de reglas construyen un grafo dirigido en el que los nodos son los hechos (iniciales, intermedios y conclusiones) y las reglas los arcos, de forma que existirá un arco si y sólo si existe una regla que ligue ambos hechos/nodos.
A partir del grafo se calculan las matrices de adyacencia y de la traza de la matriz se analiza si ésta indica la existencia de caminos que no se debería dar, se pueden detectar redundancias, circularidad, etc.

7.1.2. Validación

En el proceso de **validación** es deseable que se involucre un amplio perfil de colaboradores, no sólo los ingenieros de conocimiento desarrolladores del sistema, sino también los expertos y uno o varios de los usuarios finales, cuya opinión también es importante, casi tanto como la de los propios expertos si son ellos quienes lo van a usar.

Como ya se ha mencionado, no sólo hay que validar el resultado final (que, en el peor de los casos, puede ser espúreo), sino si cada paso, paso intermedio, consejo, etc. que da el sistema lo hace de forma correcta. También es clave que la forma de razonamiento sea la correcta, es decir, que el camino que sigue el sistema para llegar a una conclusión sea el mismo que utiliza el experto (a no ser que esté justificado, por ejemplo, porque la forma proceder actual sea incorrecta), pues lo hace más usable. Asimismo es importante respetar el tipo de lenguaje que se usa, las expresiones, e incluso cuestiones como que si el usuario no entiende algo que le comunica el sistema (por ejemplo, una pregunta), que éste pueda rehacer el mensaje de otra manera. Y, por supuesto, evitar mensajes de error alarmantes, facilitar la recuperación de sesiones y asegurar los pasos que se dan mediante ventanas de confirmación.

Hay que ser cuidadosos en cómo se realiza la validación: debe comunicarse al personal involucrado cuánto se estima que va a durar (intervalo temporal realista), la metodología y pasos que se deberán seguir, así como las expectativas del proceso, evitando en la medida de lo posible que se convierta en una tarea tediosa.

Referencias estándar de los SBC

El problema de la validación es que hay que hacerla con respecto a una referencia. Tanto SBC como SSEE no se dedican a cosas cuyo resultado se conozca siempre perfectamente si es correcto o no, es decir, la necesaria **referencia estándar** (también llamada en ocasiones *regla de Oro*) puede no existir, o bien existir una referencia pero no ser completamente estándar, como sucede si la referencia es el experto o un grupo de expertos, que pueden no estar de acuerdo y ¿cómo se sabe quién tiene razón? La compensación que tiene disponer de varios expertos (que deberían ser siempre de un mismo "nivel") es que se puede clasificar el sistema con respecto al tipo de expertos al que se ajusta, siendo, obviamente, preferible que no se ajuste sólo a aquél que más ayudó al desarrollo del mismo.

En caso de definir un estándar, debe ser realista, no se puede pretender que el sistema vaya más allá de unos determinados niveles.

Para eliminar sesgos y desviaciones suelen ser útiles los **estudios ciegos**, en los que los expertos dan su opinión sobre respuestas del propio sistema y otros colegas, sin identificar de quién provienen, ante un problema. Sea como fuere, en toda validación es necesario efectuar más de una iteración.

Puede suceder que haya en el sistema/contexto/problema variables más importantes que otras, que necesitamos con más urgencia que funcionen bien, lo que puede establecernos una secuencia de validación por prioridades, hacia el ajuste fino.

Otro aspecto clave es mantener los problemas relacionados con la *usabilidad* al margen de los problemas identificados en la validación, esto es, relativos a las bases de conocimiento.

Métodos de validación

Existen dos grandes tipos de métodos de validación:

Métodos cualitativos

- Validación retrospectiva (contra casos históricos).
- Validación prospectiva (contra casos del día a día).
- Test de Turing (validaciones anónimas en varias iteraciones).
- Validación de subsistemas semiindependientes.

Métodos cuantitativos (fundamentalmente métodos estadísticos)

- Tanto por ciento de acuerdo.
- Índice kappa.

Apéndice A

Ampliaciones

En este apéndice se reproducen algunas de las figuras que intercalan el texto de los capítulos de estos apuntes, en un mayor nivel de detalle.

A.1. Figuras

Detalle de la figura 1.1 (página 4):

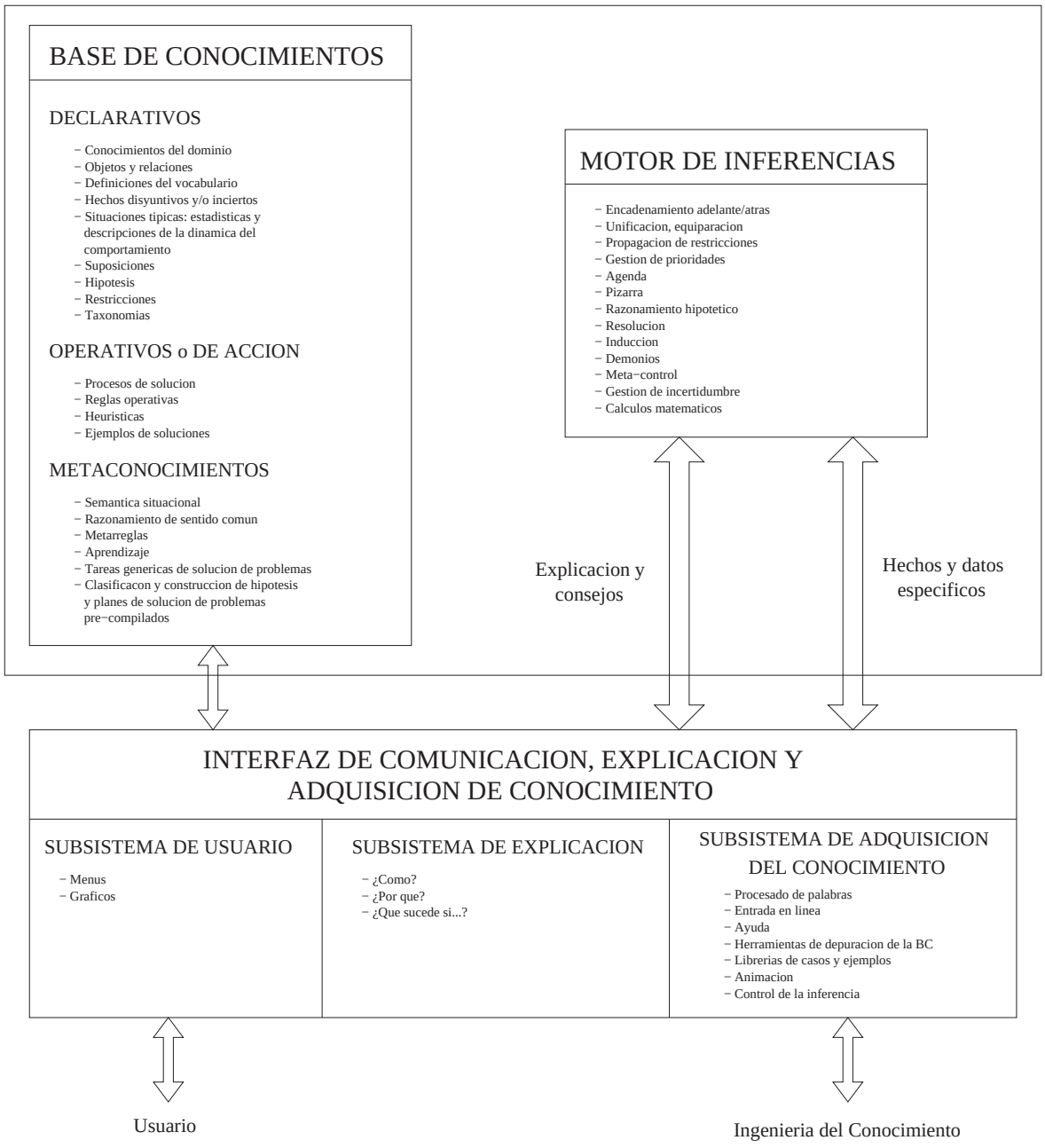


Figura A.1: Esquema detallado de un SBC.

Índice de cuadros

Índice de figuras

Bibliografía

[1] Alonso Betanzos, Amparo. *Apuntes de clase.*

[2] Vicente Moret, Amparo Alonso, et al. *Fundamentos de Inteligencia Artificial.* Servicio de Publicaciones de la Universidad de La Coruña, 2000.

Índice alfabético