CommonKADS y el Lenguaje de Modelado Unificado – UML

Pedro Salcedo Lagos
Departamento de Metodología de la Investigación e Informática Educativa,
Facultad de Educación, Universidad de Concepción
CHILE - psalcedo@udec.cl

Resumen

En este trabajo se exponen brevemente los argumentos que refrendan el Modelo de Pericia y la tricotomía *tarea/método/dominio* propia de las metodologías más difundidas en Inteligencia Artificial (IA) y específicamente CommonKADS, y se plantea el enfoque orientado a objetos (OO) como alternativa, se defiende la adecuación de las metodologías OO, frente a las metodologías funcionales, para el modelado del conocimiento experto. Llegando a establecer finalmente la correspondencia entre los modelos de Pericia de CommonKADS y los de UML, relacionando así los conceptos de los respectivos metamodelos en los niveles de dominio, tarea y método.

Palabras Claves: CommonKADS, UML, Metamodelo KADS, Metamodelo UML, Ingeniería del Conocimiento.

1. Ingeniería del Conocimiento

La ingeniería del conocimiento (IC) es un conjunto de conocimientos y técnicas que permiten aplicar el saber científico a la utilización del conocimiento (entendimiento, inteligencia o razón natural). Esta disciplina moderna puede ayudar a construir aplicaciones y sistemas orientados al aprendizaje apoyándonos con metodologías instruccionales y con tecnologías de computación y de telecomunicaciones.

Haciendo uso de las técnicas y herramientas de la IC podemos diseñar, desarrollar, producir y administrar los ambientes de aprendizaje que demandan actualmente nuestras empresas e instituciones educativas complementando de esta manera a la modalidad presencial ya en uso.

La IC posibilita la construcción de productos del aprendizaje tales como cursos, talleres, programas educativos, etc.; de manera interactiva, no lineal y a distancia; en las modalidades semi-virtual, virtual y colaborativa.

Entre las metodologías con que cuenta la IC se destaca CommonKADS como el estándar europeo para el desarrollo de sistemas basados en el conocimiento. Esta metodología es el resultado del proyecto ESPRIT KADS-II (P5248) que es una continuación del proyecto KADS. Cubre todos los aspectos del desarrollo de un SBC (conocimiento estratégico, gestión del proyecto, integración, adquisición del conocimiento y desarrollo del producto) enmarcados en un único ciclo de vida de carácter espiral, que llega incluso a la definición del programa que finalmente será ejecutado. KADS-I (ESPRIT-I P1098) se quedaba en la definición del modelo conceptual. Una de las principales contribuciones de este proyecto es el introducir las últimas técnicas aplicadas en la ingeniería del software en el campo de la IA.

2. CommonKADS

CommonKADS [1] es una metodología diseñada para el análisis y la construcción de sistemas basados en conocimiento (SBC) de forma análoga a los métodos empleados en ingeniería de software. Fue propuesta y desarrollada por un grupo de investigadores pertenecientes a diversos países de la comunidad Europea, a través del programa ESPRIT para la innovación y la aplicación de Tecnología Informática avanzada. El trabajo se comenzó en 1983 cuando había poco interés en tales metodologías. En ese momento, la construcción de sistemas de conocimiento estaba enmarcada bajo el paradigma de desarrollo por prototipos y de representación del conocimiento a través de reglas de producción, con hardware y software de propósito especial como máquinas LISP y PROLOG, herramientas especiales para sistemas expertos, etc.

Lo que se pretendía era crear un estándar para ingeniería del conocimiento y sistemas de conocimiento con el cual se pudieran construir sistemas industriales de calidad a gran escala, en una forma estructurada y controlada.

En el desarrollo de CommonKADS han participado investigadores de diferentes áreas, de diferentes universidades europeas, e incluso empresas que han servido para ver su aplicación y validar lo establecido. Sobre esta metodología se han presentado varios artículos y ponencias en revistas y eventos especializados y se han escrito algunos libros, con el fin de que se conozca y aplique en la solución de problemas reales.

A pesar de que el proyecto terminó en 1994, se han seguido desarrollado investigaciones alrededor de CommonKADS. Esto se ha logrado mediante el desarrollo de tesis doctorales que le han adicionado funcionalidad a la metodología, como por

ejemplo algunas propuestas para comenzar a generar métodos de solución de problemas para el dominio del diagnóstico [2]. Algunas son para añadirle características que permitan que la metodología se utilice en el desarrollo de sistemas multiagentes [3] o para sistemas inteligentes en tiempo real [4].

El desarrollo de esta metodología ha sido financiado por la Comunidad Europea entre 1983 y 1994 a través de varios proyectos.

En CommonKADS podemos ver reflejadas tres ideas que han emergido, no solo de la experiencia en la Ingeniería del conocimiento, sino también en del campo de la Ingeniería del Software en general. Estas tres ideas se pueden concretar en tres conceptos: modelado, reutilización y gestión del riesgo.

La metodología *CommonKADS* abarca todo el ciclo de desarrollo de software (puesto que se extiende no solamente a SBCs sino al software en general) mediante un número de modelos interrelacionados que capturan los principales rasgos del sistema y de su entorno.

El proceso de desarrollo de SBC consiste en rellenar un conjunto de "plantillas" de los modelos. Asociados a estas plantillas, *CommonKADS* define "estados" de los modelos que caracterizan hitos en el desarrollo de cada modelo. Estos estados permiten la gestión del proyecto, cuyo desarrollo se realiza de una forma cíclica dirigida por los riesgos.

Hay seis modelos definidos en CommonKADS (Figura 1)

- *Modelo de la Organización (OM)*: es una herramienta para analizar la organización en que el SBC va a ser introducido, y pretende descubrir problemas y oportunidades.

- *Modelo de Tarea (TM) (Subpartes relevantes del proceso):* describe a un nivel general las tareas que son realizadas o serán realizadas en el entorno organizativo en que se propone instalar el SBC y proporciona el marco para la distribución de tareas entre agentes.

- *Modelo de Agente (AM)*: un agente es un ejecutor de una tarea. Puede ser humano, software o cualquier otra entidad capaz de realizar una tarea. Este modelo describe las competencias, características, autoridad y restricciones para actuar de los agentes.

- *Modelo de Comunicaciones (CM)*: detalla el intercambio de información entre los diferentes agentes involucrados en la ejecución de las tareas descritas en el modelo de tarea.

- *Modelo del Conocimiento (de Pericia o de Experiencia - EM):* este es el corazón de la metodología CommonKADS y modela el conocimiento de resolución de problemas empleado por un agente para realizar una tarea. El modelo de la experiencia distingue entre el conocimiento de la aplicación y el conocimiento de resolución del problema. El conocimiento de la aplicación se divide en tres subniveles: nivel del dominio (conocimiento declarativo sobre el dominio), nivel de inferencia (una biblioteca de estructuras genéricas de inferencia) y nivel de tarea (orden de las inferencias).

- *Modelo de Diseño (DM):* mientras que los otros cinco modelos tratan del análisis del SBC, este modelo se utiliza para describir la arquitectura y el diseño técnico del SBC como paso previo a su implementación. En general produce la especificación técnica en términos de arquitectura, plataforma de implementación, módulos de software, construcciones de representación, y mecanismos computacionales para la implementación de SC.

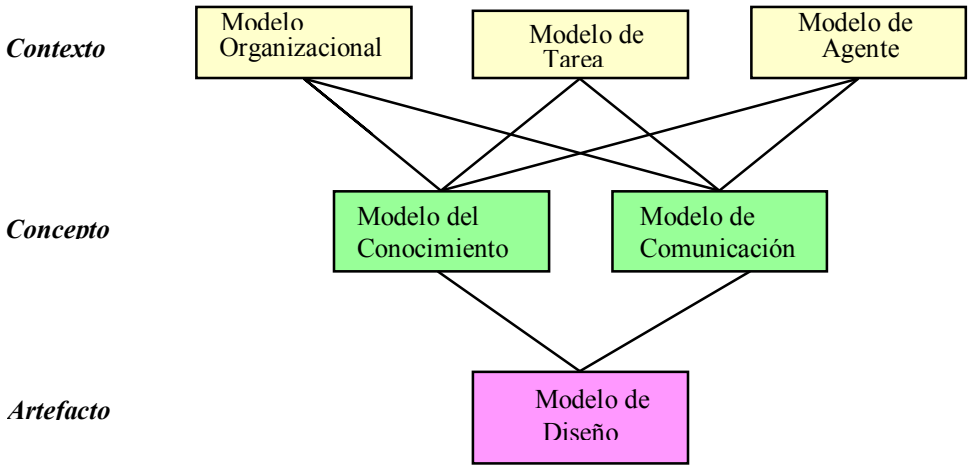

Fig. 1 "Modelos definidos por CommonKADS"

El principal producto que resulta de la aplicación de CommonKADS son estos modelos, los que se puede considerar como una agrupación estructurada de conocimiento que refleja todos aquellos aspectos importantes para que el SBC tenga éxito dentro de un contexto organizacional determinado.

Para los tres primeros modelos, del contexto, es posible utilizar 9 formularios o plantillas propuestas por CommonKADS [5]:

- modelo organizacional: 5 formularios.
  - OM-1. Identificación del problema / oportunidad (lista de problema/oportunidades percibidas, contexto organizacional [misión-visión-objetivos de la organización], lista de posibles soluciones).
  - OM-2. Aspectos Variantes (describe los aspectos que podrían cambiar o ser afectados por la solución dada por un sistema de conocimiento).
  - OM-3. Proceso de la organización dividida en partes (llenado para cada tarea de la descripción del proceso).
  - OM-4. Activos de conocimiento (detalle del elemento conocimiento de OM-2, descripción con granulo grueso, refinado en el modelo tarea y modelo conocimiento).
  - OM-5. Factibilidad (beneficios esperados, valor agregado esperado, costos esperados, comparación entre posibles soluciones, cambios organizacionales requeridos, riesgos e incertidumbres económicas y de negocios).

- modelo de tareas: 2 formularios (subporción de un proceso de la organización, actividad que agrega-valor de manera dirigida por metas).
    - TM-1. Análisis y descripción de la tarea dentro del proceso.
    - TM-2. Elementos de conocimiento de la tarea.
- modelo de agentes: 1 formulario (perspectiva proceso/tarea, perspectiva de agentes individuales).
    - AM-1. Formulario Agente
- resumen: 1 formulario.

Los modelos de experiencia y agentes proporcionan los requisitos de entrada que guiarán la implementación del sistema a través del modelo de diseño. Como se puede deducir, cada una de estas agrupaciones intentan responder a cada una de las preguntas claves en el desarrollo de un SBC: ¿Es un SBC la solución idónea para el problema que se quiere resolver?, pregunta que se puede responder por la descripción del contexto dado por los tres primeros modelos; ¿Cuál es la naturaleza y la estructura tanto del conocimiento como de la comunicación utilizada?, cuya respuesta se puede encontrar en el modelo de experiencia y en el modelo de comunicación; y finalmente, ¿Cómo debe ser implementado el conocimiento?, pregunta que intenta esclarecer el modelo del diseño.

Mención especial al modelo de conocimiento. Este modelo, que está descrito en [6], describe el conocimiento que tiene un determinado agente y que es relevante para la consecución de una determinada tarea, además de describir la estructura del mismo en función de su uso. Obviamente, este modelo se hace en el nivel de conocimiento, sin hacer referencia a aspectos de implementación. Para poder llevar a cabo este modelado de los distintos papeles que puede jugar el conocimiento, éste está distribuido en tres categorías:

- **Conocimiento de tareas**. Describe de una forma recursiva la descomposición de una tarea de alto nivel en varias subtareas. El conocimiento sobre una tarea se divide en dos partes: por una lado la tarea, que sirve para especificar que es lo que implica la aplicación de la tarea ya que define su objetivo en términos de los roles de entrada y de salida; por otro lado, está el método de la tarea, que define el cómo se lleva a cabo dicha tarea, indicando en qué subtareas se descompone y en qué orden deben de ser procesadas (control).
- **Conocimiento del dominio**: que se compone a su vez de
    - **Ontologías del Dominio**: que proporcionan el vocabulario de las entidades del dominio, sus relaciones, y las restricciones en su estructura. Se pueden ver como metamodelos del conocimiento del dominio. Por ejemplo, el Frame Ontology [8] define 63 meta-predicados que se pueden utilizar para caracterizar clases, relaciones, atributos e instancias.
    - **Modelos del dominio**: que describen el conocimiento sobre el dominio en particular. Consiste en conjuntos de tuplas formuladas en el vocabulario definido en la ontología del dominio y que satisfacen sus restricciones. Muestran relaciones entre diferentes elementos de conocimiento.
      Conceptos: Clases de objetos, abstracciones o del mundo real, representando objetos físicos o estados.
      Propiedades: Atributos de los conceptos.

Expresiones: Afirmaciones del tipo "la propiedad del concepto tiene el valor"

Relaciones: Conexiones entre cualesquiera elementos del dominio, esto es, que un concepto afecta de alguna manera a otro.

- **Conocimiento sobre inferencias**. Describe los procesos primitivos de razonamiento que tienen lugar en una aplicación, así como los roles de conocimiento que son usados por las inferencias. Obviamente, estos roles de conocimientos están relacionados con elementos del conocimiento del dominio. Hay que tener en cuenta, que las inferencias son consideradas primitivas respecto a un modelo de experiencia determinado, ya que en otros modelos de experiencia la misma inferencia puede ser una tarea descomponible.

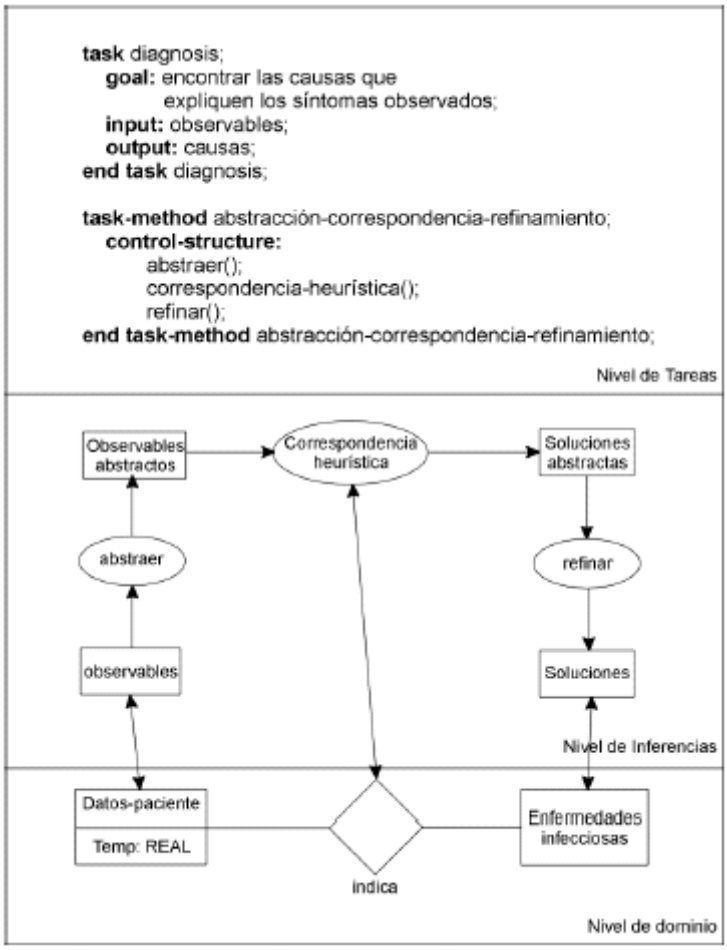

Fig. 2 "El modelo de Conocimiento en CommonKADS"

CommonKADS propone el lenguaje CML (Conceptual Modelling Language) [8], para materializar la especificación del modelo de conocimiento. Este lenguaje permite la definición de todos los elementos anteriormente citados, así como la estructuración del conocimiento en las partes anteriormente citadas. Además propone el uso de una notación gráfica, que permite no sólo la definición de las estructuras de tareas (relación tareas-subtareas), sino que también permite la definición de la ontología y los conceptos del dominio y la definición de la dependencia de los datos entre las inferencias a través de las estructuras de inferencias (Figura 2). Aparte de CML, también se propone el uso de $(ML)^2$ [9] que está más orientado hacia la formalización del modelo. Esta descomposición del nivel de conocimiento en el conocimiento específico del dominio por un lado, y de las tareas y las inferencias por otro, favorece la

reutilización del conocimiento en dos niveles. Al haber definido un conjunto de inferencias elementales independientes de la aplicación se permite que estas sean utilizadas en otros procesos de resolución de problemas. En este sentido es importante tener en cuenta el trabajo de donde se catalogan y clasifican el conjunto de inferencias básicas que pueden ser utilizadas en CommonKADS. El otro nivel de reutilización lo establece el conocimiento del dominio, que al ser definido independientemente de las tareas, puede ser reutilizado por otro conjunto de tareas definidas sobre el mismo dominio. Para favorecer la reutilización del conocimiento del dominio CommonKADS permite definir las ontologías a distintos niveles de generalización y distintos puntos de vista, lo que abre el abanico de posibles adaptaciones a otras aplicaciones.

Otro de los aspectos importantes que introdujo CommonKADS fue la definición de un marco de trabajo para la gestión y planificación del proyecto. CommonKADS define un ciclo de vida para el desarrollo del proyecto basado en un modelo en espiral como el propuesto por [10]. El modelo en espiral que plantea CommonKADS se basa en los siguientes principios [5]:

- La planificación del proyecto que se centra principalmente en los productos y las salidas que tienen que producirse como resultado, más que un conjunto de actividades o fases.
- La planificación se realiza de una forma adaptativa a lo largo de un serie de ciclos en espiral, que están dirigidos por una valoración sistemática de los riesgos del proyecto.
- El control de calidad es una parte más de la gestión del proyecto, ya que la calidad está integrada en el desarrollo del SBC por medio de la metodología.

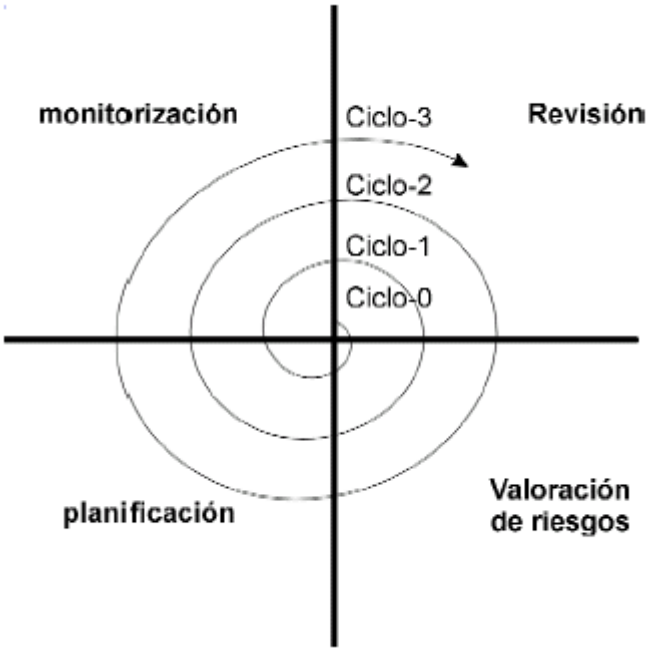

Fig. 3 "El ciclo de vida de CommonKADS"

Estos principios están garantizados por un lado, por el conjunto de modelos, y por otro, por el ciclo de vida en espiral. Este ciclo de vida consta de cuatro fases (Figura 3):

- **Revisión**. Es el primer paso de cada ciclo y en el se revisa el estado actual del proyecto y se establecen los objetivos principales que se quieren cubrir en el ciclo en cuestión.

- **Valoración de riesgos**. Las líneas generales del proyecto establecidas en el paso anterior sirven de entradas para esta fase. Su función principal es la identificación y valoración de los principales obstáculos que nos podemos encontrar para la consecución exitosa del proyecto, así como las acciones que se deben tomar para minimizar dichos riesgos.
- **Planificación**. Una vez obtenida una visión clara de los objetivos que hay que cubrir, los riesgos que se pueden presentar y las acciones que hay que tomar, hay que realizar una planificación del trabajo a realizar. En dicha planificación hay que establecer la distribución de la carga del trabajo en términos de qué tareas hay que realizar, una temporalización de dichas tareas, la distribución de los recursos, etc.
- **Monitorización**. Es la última fase del ciclo y está constituida por el desarrollo propiamente dicho. El trabajo realizado en esta fase está controlado y dirigido por el director del proyecto. Para determinar el grado de cumplimiento de los objetivos se requieren reuniones con los agentes implicados en el proyecto (usuarios, administradores, expertos, ..). El resultado de dichas reuniones se utilizará como entrada del proceso de revisión del siguiente ciclo.

Como se puede observar, la metodología CommonKADS abarca todo los aspectos del desarrollo de un SBC, desde los análisis iniciales que sirven para identificar problemas y para establecer la idoneidad de la solución basada en un SBC, hasta la implementación del mismo, proporcionando un marco de trabajo donde llevar a cabo la gestión del proyecto. También hay que resaltar que el modelado del conocimiento posibilita la definición de componentes reutilizables, tanto en el nivel de tareas como en el de conceptualización del dominio, como resultado de la agrupación de las ideas que surgieron después del trabajo de Newell [Newell, 1982]. Todo esto hace que CommonKADS haya sido adoptado no sólo en Europa, donde se ha convertido en un estándar de facto, sino que también empieza a ser utilizado en el resto del mudo.

Si bien KADS-I veía el proceso de construcción del modelo como una interpretación del problema en términos de un modelo preestablecido (modelos de interpretación). CommonKADS ofrece dos visiones alternativas (o complementarias). La modificación o adaptación del modelo preexistente [11] o el análisis de la tarea de acuerdo a la estructura tarea-método-subtareas [12]. En cualquier caso es imprescindible disponer de una biblioteca de componentes reutilizables. En los últimos años se ha trabajado mucho en la definición de un marco, dentro de CommonKADS, para la definición de bibliotecas de componentes reusables [13].

En la biblioteca, desde un punto de vista constructivista [14], se distinguen tres tipos de componentes:

Componentes de modelado: particiones de un modelo de pericia en elementos que no solapan sus contenidos. Proporcionan la adecuada flexibilidad para su recombinación. Son conocidas de antemano sus interconexiones, en términos de contenidos y papel que juegan en el modelado.

Área de tareas:

Funciones que representan la visión funcional de los pasos de razonamiento. En la biblioteca no se diferencia entre funciones pertenecientes a la capa de inferencias o tareas. Esta distinción es propia de su utilización en un modelo, inferencia primitiva o subarea.

Estructuras funciona1es, grupos de funciones que representan descomposiciones de funciones y sus dependencias a través de los roles que el conocimiento juega en la resolución del problema.
Estructuras de control que representan la secuencia de control necesaria en descomposición para hacerla operativa.

Área del dominio.

Ontologías del dominio que representan la terminología de un dominio de una manera coherente y estructurada.

Modelos genéricos del dominio que representan estructuras reusables del conocimiento del domino para el proceso de resolución del problema. El término genérico se utiliza para diferenciar los modelos del domino en la biblioteca de aquellos usados en 1a aplicación.

Se consideran dos tipos de conexiones entre estos componentes: los PSMs, que a su vez pueden ser métodos de expansión (descomposición) y métodos de control. Y capacidades (features), esto es, los compromisos ontológicos de PSM.

Modelos genéricos: marcos que representan una clase de modelos de pericia completos, en la línea de las tareas genéricas de Chandrasekaran y los modelos de interpretación de KADS.
Operadores de modelado: Relaciones entre modelos genéricos que representan un paso potencial en el modelado. Toman componentes de modelado como parámetros para transformación parcial del modelo de pericia. Realmente este concepto no se ha popularizado en aplicaciones (ejemplos en [15]) y menos como herramientas automatizadas.

Todos estos componentes se indexan básicamente de dos formas: por capacidades o mediante tipologías establecidas (Fig.4). En el primer caso, recuperar y rehusar un componente o modelo genérico de la biblioteca exige el cumplimiento con ciertas restricciones y suposiciones asociadas intrínsecamente al componente. Por ejemplo, para usar un método "cubre&discrimina" (su función asociada) en una aplicación específica, se exige identificar explicaciones formadas por rutas causales.

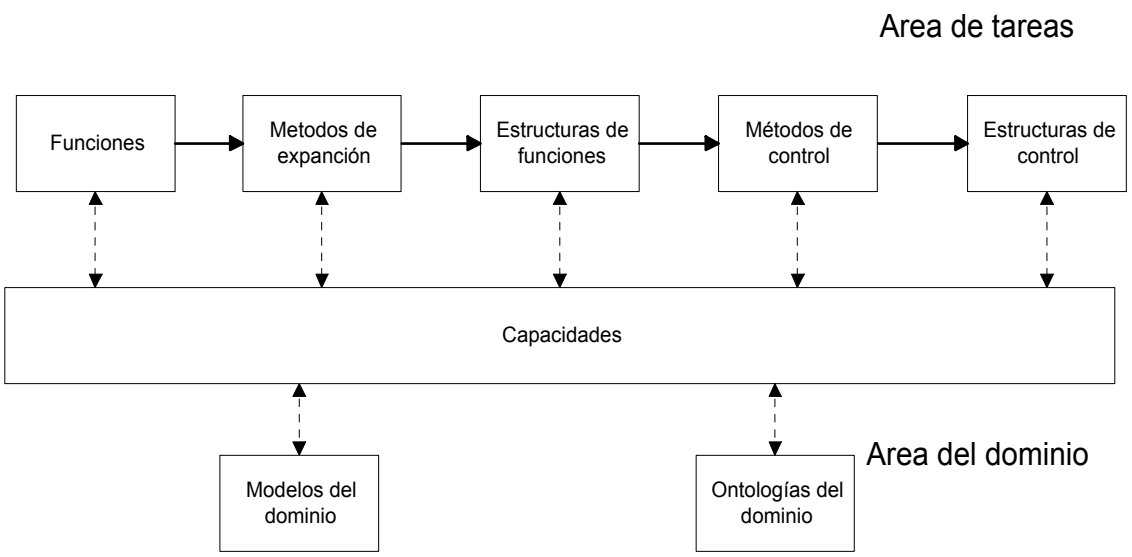

Fig. 4 "Estructura de la biblioteca"

Estructura de la biblioteca: componentes del modelo de pericia y sus conexiones: con línea continua las conexiones estructurales y con línea discontinua las capacidades de los componentes.

En KADS-I se establecían básicamente dos tipologías de componentes: tareas genéricas o modelos de interpretación, e inferencias. En la misma línea, en CommonKADS se establecen clasificaciones de:

Funciones canónicas: Que contienen funciones que denotan operaciones como abstract, select, specify, match, compute, transform, etc., (en realidad son los verbos inferenciales o fuentes de conocimiento de KADS-I [16]). Es una propuesta de conjunto de primitivas ontológicas del razonamiento, que ayudan a identificar los pasos de razonamiento. La tipología se fundamenta en simples esquemas de razonamiento sobre los elementos de la ontología (conceptos, relaciones etc.) Provienen de las operaciones sobre las primitivas de lenguaje KL-ONE: concepto, atributo, valor, instancia, conjunto y estructura. Tienen la ventaja de usar términos estandarizados sobre estructuras estandarizadas. Ayudan al ingeniero del conocimiento al proporcionar un conjunto de verbos que puede utilizar y además también le ayudan a buscar las estructuras en el dominio adecuadas para tal verbo. Son los elementos semánticos terminales.

Tipos de problemas: Tipos primitivos de problemas: diagnóstico, diseño, planificación etc. Se asume que las expansiones de las funciones que solucionen estos problemas son típicas o incluso exclusivas [13]. Por ejemplo los métodos de diagnóstico son típicos de tareas de diagnóstico. Cuando se aplica un método en un tipo de problema, supone añadir a éste un apellido: Diagnóstico sistemático, diagnóstico heurístico, etc. Pero la tipología en sí, no pretende ser una taxonomía. La caracterización de los problemas viene de su objetivo, de lo que se persigue como solución del problema. Desde luego, supone una abstracción que parte de razonamientos análogos al siguiente "algo tienen que tener en común la localización de una pieza defectuosa, la clasificación de una disfunción o la búsqueda de una explicación o encadenamiento causal que ha producido un accidente, aunque la forma de proceder sea muy diferente".

En [15] podemos encontrar descomposiciones, según diferentes autores, de otras tareas (o tipos de problemas, según su denominación) en la misma línea de Benjamins:

- Diagnóstico basado en modelo (Bredeweg)
- Predicción de comportamiento (Breuker y Van der Velde)
- Evaluación (o valoración) (Valente y Löckenhoff)
- Diseño (Bernaras y Van de Velde)
- Configuración (Löckenhoff y Messer)
- Planificación (Valente)
- Asignación y Planificación temporal (Sundin)
- Modelado (Top y Akkermans)

En [15] se propone una interrelación entre los tipos de problemas. Se propone un punto de vista para su organización a partir de la dependencia entre ellas, por ejemplo el diagnóstico basado en un modelo requiere el diseño de dicho modelo, y éste es el resultado de una tarea de diseño.

3. Notaciones UML en CommonKADS.

Correspondencia entre los elementos de CommonKADS y UML

El Lenguaje Unificado de Modelado, en adelante UML (Unified Modeling Languaje), es el resultado mas integrador de una serie de métodos de análisis y diseño orientado a objetos [17] [18] [19].

Originado entre fines de los ochenta y principios de los noventa, UML no fue concebido como un método en sí mismo, sino como la notación básicamente gráfica de la que se puede valer cualquier método para expresar los diseños y el proceso que orienta los pasos a dar para realizar este diseño. Así completa lo que todo método debe presentar, un lenguaje de modelado y un proceso. Al proceso en sí, se le ha llamado Método Unificado (Unified Method u Objectory).

Sucesor de las notaciones OMT, Booch y OOSE, pronto constituirá un estándar de facto en el campo del análisis orientado a objetos, posición consolidada por la estandarización de OMG [Object Management Group – 1997].

Al contrario que sus predecesores, UML posee una semántica bien definida lo que hace posible la integración de la tecnología de verificación y validación formal, tecnología que precisa descripciones formales de los sistemas, en los ciclos de desarrollo software orientado a objetos. Permite pues la definición de modelos sin ambigüedad cuya completitud y consistencia semántica pueden ser comprobadas con el apoyo de herramientas software diseñadas al respecto, que asimismo pueden ofrecer la propia ejecución, validación y simulación automática de los modelos. La posibilidad de conectar dichos modelos con diversos lenguajes de programación (tales como Java, C++ o Visual Basic) permite también el diseño de herramientas para la generación automática de código y recíprocamente, para soportar procesos de ingeniería inversa (es el caso de la herramienta Rational Rose). UML promueve particularmente procesos de análisis iterativos, increméntales y en diferentes niveles de abstracción. De este modo resulta muy útil para definir modelos genéricos que se especializan en dominios concretos.

Aunque la semántica del UML no es del todo formal, su sintaxis sí se basa en un metamodelo formalmente definido. Es pues un lenguaje de carácter semi-formal, bastante flexible y fácil de entender y manipular, al contrario que otros lenguajes estrictamente formales, a la vez que permite mayor precisión y rigor que el lenguaje natural. Semánticamente, es también mucho más rico que el código, y permite incorporar la especificación de detalles que no se traducirán a la implementación. De este modo, se conserva una gran cantidad de información de la especificación inicial que de otra forma se perdería, información inestable para la interpretación inequívoca de un código derivado. Adicionalmente, sus notaciones gráficas semiformales se asemejan a otras de uso ya muy extendido para el modelado de sistemas reales de granescala.

El lenguaje UML permite describir la arquitectura de los sistemas software mediante cinco visiones entrelazadas, cada una de las cuales es una proyección de la organización y estructura del sistema que focaliza en un aspecto particular del mismo. Estas cinco visiones son: la visión de casos de uso, la visión de diseño, la visión de procesos, la visión de implementación y la visión de despliegue. Desde el punto de vista del modelado en un alto nivel de análisis interesan especialmente las dos primeras

visiones, adquiriendo relevancia también la tercera cuando entran en consideración los métodos cooperativos.

El metamodelo de UML está constituido principalmente por elementos estructurales o estáticos (para un propósito de análisis interesan los elementos conceptuales o lógicos: clases, colaboraciones, casos de uso y clases activas), elementos conductuales o dinámicos (interacciones y máquinas de estado), relaciones o conexiones entre los diferentes elementos (dependencia, asociación, generalización y realización), elementos de agrupación y elementos para anotaciones.

Los **Diagramas de Clases** en UML (Fig. 5)

Captura la estructura estática de la información, muestra el conjunto de clases y objetos importantes que hacen parte de un sistema, junto con las relaciones existentes entre estas clases y objetos. Muestra de una manera estática la estructura de información del sistema y la visibilidad que tiene cada una de las clases, dada por sus relaciones con las demás en el modelo.

- Generalización, herencia & re-uso son tópicos importantes.
- Importados a la notación del **esquema del dominio de CommonKADS**
  o no se hace uso de la caja operación.
- Se puede usar también en el **Modelo Tarea de CommonKADS** para dibujar la estructura de la información de la tarea.

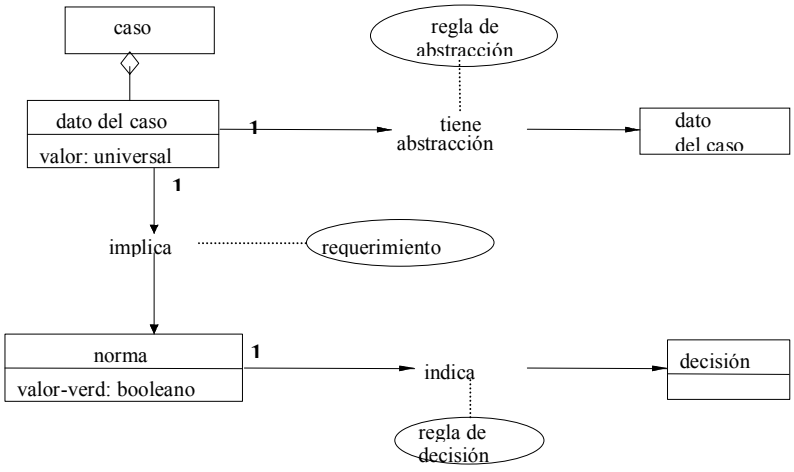

Fig. 5 Ejemplo diagrama de clase en UML

Los **Diagramas de Estado** en UML (Fig. 6)

Muestra el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación, junto con los cambios que permiten pasar de un estado a otro. Un estado se representa como un rectángulo de vértices redondeados, y una transición entre estados como una línea que los conecta.

- Sinónimos: “carta de estado”, “diagrama de transición de estados”
- Propósito: modelar el comportamiento dinámico
- Usar si el control es influenciado fuertemente por eventos “externos”

- Dibujar un diagrama de estado para clases de objetos con comportamiento interesante
- Usado en CommonKADS en el **Modelo de Comunicación**

Los **Diagramas de Actividad** en UML (Fig. 7)

Un diagrama de actividades es un caso especial de un diagrama de estados en el cual casi todos los estados son estados de acción (identifican que acción se ejecuta al estar en él) y casi todas las transiciones son enviadas al terminar la acción ejecutada en el estado anterior. Puede dar detalle a un caso de uso, un objeto o un mensaje en un objeto. Sirven para representar transiciones internas, sin hacer mucho énfasis en transiciones o eventos externos. Y en general muestra los pasos y puntos de decisión que suceden *dentro del comportamiento de un objeto, o dentro de un proceso de negocios*. Cada paso es un rectángulo de vértices redondeados (de una forma más ovalada que la representación de un estado) y cada punto de decisión es un rombo.

- Modela el flujo de control e información de un procedimiento o proceso.
- Útil si el control es principalmente sicrónico
- Usado **en CommonKADS para modelar el proceso organizacional** (formulario OM-2 del modelo de organización).
- Se puede usar también **en CommonKADS para modelar el flujo de control en un método tarea** (Modelo del Conocimiento).

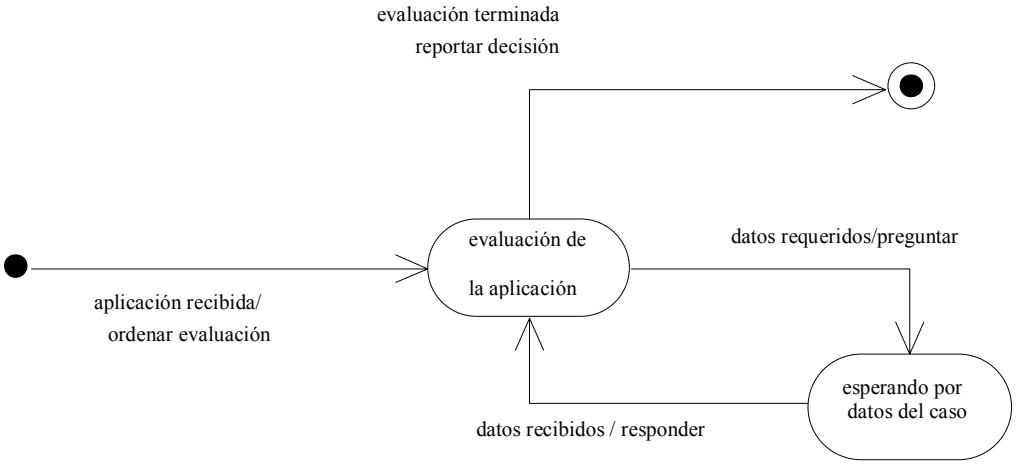

Fig. 6 Ejemplo Diagrama de Estado en UML

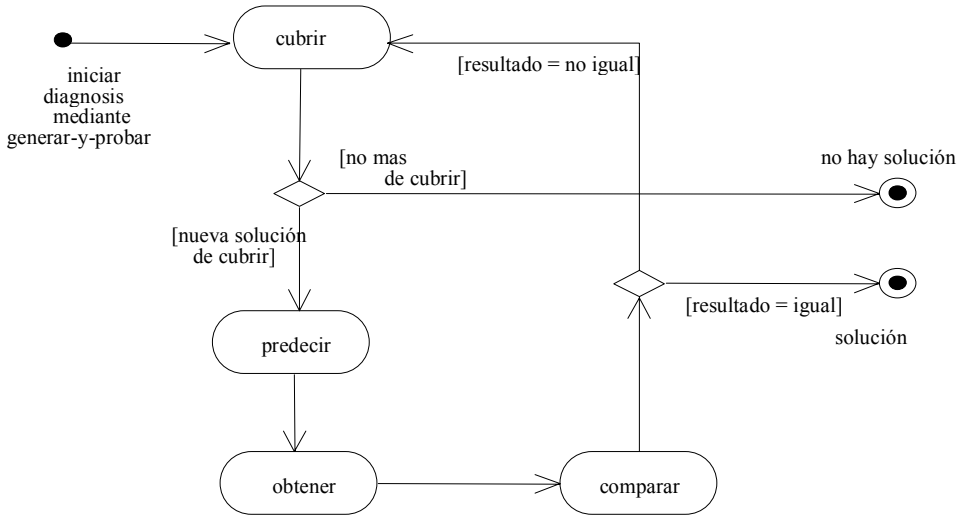

Fig 7 Diagrama de Actividad del Método Control en UML

Los **Diagramas de Casos de Uso** en UML (Fig. 8)

Un diagrama de Casos de Uso muestra las distintas operaciones que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones).

- Muestra los servicios que se pueden recibir de un sistema
- provée una vista externa (cliente)
- terminología
  - o *use case (servicio provísto por el sistema)*
  - o *actor (agente usando los servicios de un sistema)*
- usado en las fases iniciales del análisis del sistema
- usado **en CommonKADS** como una manera de presentar las soluciones a los clientes (**propuesta y proceso organización**)

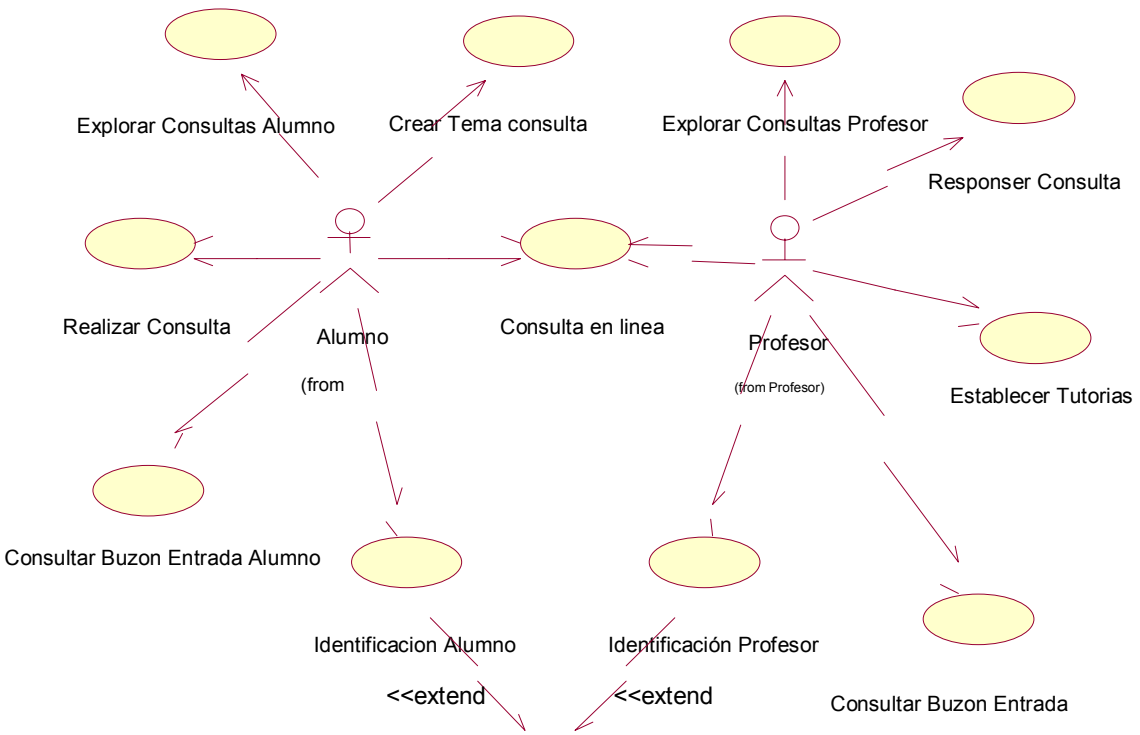

Fig 8 Diagrama de Casos de Uso en UML

Conclusiones

Aunque UML no es del todo formal, su flexibilidad, la facilidad de uso, su mayor precisión y rigor frente al lenguaje natural, su riqueza semántica frente al código, la posibilidad de incorporar más detalle en la especificación que no se traducirán en la implementación, lo hacen un lenguaje apropiado para establecer las correspondencias entre los modelos de Pericia de CommonKADS y los de UML, relacionando así los conceptos de los respectivos metamodelos en los niveles de dominio, tarea y método. A juicio del mismo Schreiber (Booch et al, 1998) el uso de notaciones UML, se considera "similar en espíritu" a CommonKADS.

En definitiva, en este trabajo se han expuesto los argumentos que refrendan el Modelo de Pericia y la tricotomía *tarea/método/dominio* propia de las metodologías más difundidas en Inteligencia Artificial (IA) y específicamente CommonKADS, y se ha establecido la correspondencia entre los modelos de Pericia de CommonKADS y los de UML, relacionando así los conceptos de los respectivos metamodelos en los niveles de dominio, tarea y método.

Bibliografía

[1] R. de Hoog, W. Post, B. J. Wielinga and A. Th. Schreiber. **Organizational Modeling in CommonKADS: the Emergency Medical Service**. *IEEE Expert* **12***(*6), 1997.

[2] V. R. Benjamins. ***Problem Solving Methods for Diagnosis***. PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, June 1993.

[3] C.A.Iglesias., **Definición de una metodología para el desarrollo de sistemas Multiagente**. Tesis Doctoral , febrero 1998. Dpto. Ingeniería de Sistemas, Universidad Politécnica de Madrid.

[4] Henao, M. (1997). **Consideraciones metodológicas para el desarrollo de sistemas inteligentes en tiempo real**. Propuesta para tesis doctoral, Universidad Politécnica de Valencia, España.

[5] G. Schreiber, H. Akkermans, Anjo Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde y B. Wielinga. **Knowledge engineering and Management: The CommonKADS Methodology.** MIT Press, Cambridge, Mass. 1999.

[6] B.J. Wielinga, **Towards a Unification of knowledge Modellingg Approaches.** Second Generation Expert Systems , pp. 229-335. Springer-Verlag.

[7] Gruber, 1993. A translation approach to portable ontology specification Knowledge Acquisition, 5,199-220.

[8] A. Anjewier- den. CML2. Technical Report 11, University of Amsterdam, **http://www.swi.psy.uva.nl/usr/anjo/cml2**, 1997

[9] F. Van Harmelen, J. R. Balder, M. Aben, and J. M. Akkermans.**A formal language for KADS conceptual models**. Technical Report ESPRIT Project P5248 KADS-II/T1.2/TR/ECN/006/1.0, Netherlands Energy Research Centre ECN and University of Amsterdam, June 1991.

[10] B.W. Boehm, 1988. **A Spiral Model of Software Development and Enhancement. In IEEE Computer**, May 1988, pp. 61-72.

[11] G. SCHREIBER, B. WIELINGA & J. BREUKER. **KADS: A Principled Appoach to Knowlwdge- Based System Development**, pp. 93-118. Academic Press.

[12] Sisyphus-VT. **CommonKADS solution. International Journal of Human-Computer Studies**. 44.373-402.

[13] Valente, 1994b. **The CommonKADS Expertise Modeling Library**. J.BREUKER & W. Van De VELDE, eds., CommonKADS Library for Expertise Modeling. IOS Press

[14] Van de Velde, 1994 **Contructivist View on Knowledge Engineering**. En A. COHN, ed., Proceedings de ECAI'94 11th European Conference on Artificial Intelligence. John Wiley & Sons.

[15] Breuker & Van de Velde, 1994. **CommonKADS Library for Expertise Modelling Reusable Problem Solving Components**. Amsterdam, IOS Press.

[16] Breuker et al., 1987. **Model Driven knowledge acquisition; interpretation model. ESPRIT Project P1098 Deliverable D1 (task A1),** University of Amsterdam and STL Ltd., Amsterdam, The Netherlands.

[17] James Rumbaugh, Ivar Jacobson, Grady Booch: **The *Unified Modeling Language Reference Manual***, Addison-Wesley, 1998. ISBN 020130998X.

[18] Grady Booch, James Rumbaugh, Ivar Jacobson: ***The Unified Modeling Language User Guide***, Addison-Wesley, 1998. ISBN 0-201-57168-4.

[19] Rainer Burkhardt: ***UML: Unified Modeling Language***, Addison-Wesley, 1997. ISBN 3-8273-1226-4.