UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

SEDE VÍCTOR LEVI SASSO

SISTEMAS BASADOS EN EL CONOCIMIENTO

INCLUYE PRUEBAS SUMATIVAS Y PRESENTACIONES DEL CONTENIDO

DR. CARLOS ROVETTO

2019

CONTENIDO

INTRODUCCIÓN 8

1. CONCEPTOS, CARACTERÍSTICAS Y ESTRUCTURA BÁSICA DE LOS SBC 9

1.1. Definición de los SBC 9

1.1.1. Características de los SBC 9

1.1.2. Tipos de SBC 10

1.2. Componentes de los SBC 11

1.2.1. Base del conocimiento 12

1.2.1.1. Tipos de conocimiento 12

1.2.1.2. Estrategias para representar el conocimiento 12

1.2.2. Motor de inferencias 13

1.2.3. Módulo de adquisición del conocimiento 13

1.2.4. Módulo de explicación 14

1.2.5. Interfaz de usuario 14

1.3. Desarrollo de un SBC 14

1.3.1. Etapas de desarrollo de un SBC 14

1.3.1.1. Identificación del problema 14

1.3.1.2. Modo de desarrollo 15

1.3.1.2.1. Elección de lenguajes, herramientas y conchas 16

1.3.1.3. Desarrollo de un prototipo 16

1.3.1.4. Planificación de un sistema a gran escala 17

1.3.1.5. Implementación, mantenimiento y evolución 17

1.4. Aplicaciones de un SBC 17

1.4.1. Razones para utilizar un SBC 18

1.4.2. Ventajas y Limitaciones 19

2. HERRAMIENTAS PARA EL DESARROLLO DE SBC 20

2.1. Expert System Builder 20

2.1.1. Nodos 20

2.1.1.1. Attributes 21

2.1.1.2. Values 21

2.1.1.3. Conclusion 21
2.1.2. Ejemplo: Razas de perros 21
2.2. ExSys Corvid 25
2.2.1. Variables 26
2.2.2. Creación de Bloques Lógicos 27
2.2.3. Bloques de Comandos 28
2.2.4. Ejemplo: Solicitud de Préstamo 29
2.2.4.1. Paso 1: Crear el proyecto 30
2.2.4.2. Paso 2: Declarar las variables y asignar sus valores 30
2.2.4.3. Paso 3: Añadir los bloques lógicos (reglas) 34
2.2.4.4. Paso 4: Definir los bloques de comando básicos 36
2.2.4.5. Paso 5: Ejecutar el sistema 38
2.3. Prolog 38
2.3.1. Directivas de compilación 39
2.3.2. Sección de constantes 40
2.3.3. Sección de dominios 40
2.3.4. Sección de la base de datos 41
2.3.5. Sección de los predicados 42
2.3.6. Sección de cláusulas 43
2.3.7. Sección de meta u objetivo 44
2.3.8. Ejemplo: Uso de las secciones básicas de un programa en Prolog 44
2.4. AgentBuilder 46
2.5. Clips 48
2.6. Jess 50
3. INTRODUCCIÓN A LA METODOLOGÍA COMMONKADS 52
3.1. Aspectos generales de la Metodología CommonKADS. 52
3.1.1. Principios fundamentales de la Metodología CommonKADS. 53
3.1.2. Ciclo de Vida de la metodología CommonKADS. 54
3.2. Conjunto de modelos de la metodología CommonKADS. 55
3.2.1. Nivel Contextual – Modelos de Contexto. 55
3.2.2. Nivel Conceptual – Modelos Conceptuales. 55
3.2.3. Nivel Computacional – Modelo de Diseño. 56

4. MODELADO DEL NIVEL CONTEXTUAL EN COMMONKADS 57
4.1. Modelo de la Organización de la metodología CommonKADS 57
4.1.1. Descripción de los problemas y posibilidades de mejora. 57
4.1.2. Descripción de los aspectos de interés de la organización. 58
4.1.3. Descripción detallada de los procesos de interés. 59
4.1.4. Descripción del componente conocimiento. 61
4.1.5. Análisis de viabilidad. 61
4.2. Modelo de Tareas de la metodología CommonKADS. 63
4.2.1. Descripción detallada de tareas. 64
4.2.2. Especificación del conocimiento. 65
4.3. Modelo de los Agentes de la metodología CommonKADS. 66
4.3.1. Descripción de los agentes. 66
4.3.2. Documento de toma de decisiones sobre impactos y mejoras 67
5. MODELADO DEL NIVEL CONCEPTUAL EN COMMONKADS 68
5.1. Modelo de Conocimiento de la metodología CommonKADS. 68
5.1.1. Construcción del modelo de conocimiento de CommonKADS. 68
5.1.1.1. Identificación del conocimiento 68
5.1.1.2. Especificación del conocimiento 68
5.1.1.3. Refinamiento del conocimiento 69
5.1.1.4. Documentación sobre el modelo de conocimiento. 69
5.1.2. Categorías de conocimiento. 70
5.1.2.1. Conocimiento del dominio. 70
5.1.2.2. Conocimiento sobre inferencias. 70
5.1.2.3. Conocimiento sobre tareas. 70
5.1.3. Tipos de tareas proporcionadas por la librería de CommonKADS. 70
5.1.3.1. Tipos de tareas analíticas. 71
5.1.3.2. Tipos de tareas sintéticas. 71
5.2. Modelo de Comunicación de la metodología CommonKADS. 71
5.2.1. Plan de comunicaciones. 71
5.2.2. Transacciones. 72
5.2.3. Especificaciones del intercambio de información. 73
6. MODELADO DEL NIVEL COMPUTACIONAL EN COMMONKADS 75

6.1. Modelo de Diseño de la Metodología CommonKADS. 75
6.1.1. Principio Structure-Preserving Design en CommonKADS. 75
6.2. Proceso de Diseño en CommonKADS. 77
6.2.1. Diseñar la arquitectura del sistema. 77
6.2.2. Identificar la plataforma de implantación. 78
6.2.3. Especificar los componentes de la arquitectura. 79
6.2.4. Especificar la aplicación dentro de la arquitectura. 80
7. REPRESENTACIÓN DEL CONOCIMIENTO MEDIANTE LÓGICA DE PREDICADOS. 82
7.1. Representación del Conocimiento. 82
7.2. Lógica simbólica o formal. 82
7.3. Representación del conocimiento mediante lógica de predicados. 84
7.4. Inferencia y razonamiento. 84
7.5. Métodos básicos de razonamiento. 85
7.6. Reglas para transformar expresiones en cláusulas lógicas. 86
7.7. Ejemplos de representación de hechos. 86
BIBLIOGRAFÍA 88
ANEXOS 91

ÍNDICE DE FIGURAS

Figura 1. Estructura de un SBC. 11
Figura 2. Formulario para la creación de un nuevo proyecto en ES-Builder Web. 22
Figura 3. Actualización de la lista de proyectos. 22
Figura 4. Opción Edit Tree (izquierda) y vista del árbol por defecto del sistema (derecha). 23
Figura 5. Menú de opciones dentro de un nodo valor. 23
Figura 6. Nodo A, Tamaño, terminado. 24
Figura 7. Árbol terminado. 25
Figura 8. Estructura de las reglas en Exsys Corvid. 28
Figura 9. Pasos para la creación de un nuevo proyecto en Corvid. 30
Figura 10. Ventana para la creación de variables. 30
Figura 11. Creación variable Ingreso_Cliente. 31
Figura 12. Modificación del Prompt de la variable Ingreso_Cliente. 31
Figura 13. Ingreso de los valores para la variable Ingreso_Cliente 32
Figura 14. Configuración adicional variable Ingreso_Cliente 32
Figura 15. Creación variable Préstamo_Aprobado 33
Figura 16. Creación variable Préstamo_Denegado 33
Figura 17. Icono para agregar bloques lógicos. 34
Figura 18. Ingreso de la primera regla del sistema 34
Figura 19. Adición de acción THEN. 35
Figura 20. Edición regla para préstamo denegado. 35
Figura 21. Edición regla para préstamo aprobado 35
Figura 22. Finalización del proceso de añadir reglas. 36
Figura 23. Icono de los bloques de comando. 36
Figura 24. Inserción de bloque de comando Derive Conf 37
Figura 25. Inserción de comando Results 37
Figura 26. Icono para la ejecutar el sistema. 38
Figura 27. Sistema en ejecución 38
Figura 28. Ventana Project Manager de AgentBuilder - Tomado de (Acronymics Inc., 2004). 47
Figura 29. Ventana Rule Editor de AgentBuilder - Tomado de (Acronymics Inc., 2004). 47
Figura 30. IDE Clips para Windows tomado de (Riley, 2017) 49
Figura 31. Niveles CommonKADS y sus componentes. 56
Figura 32. Ejemplo de la estructura básica de un diagrama de actividades. 60
Figura 33. Ejemplo de diagrama de clases para especificación del conocimiento. 69
Figura 34. Ejemplo diagrama de diálogo. 72

ÍNDICE DE TABLAS

Tabla 1. Tipos de variables en Exsys Corvid. 26
Tabla 2. Formulario OM-1. 58
Tabla 3. Formulario OM-2 59
Tabla 4. Formulario OM-3. 61
Tabla 5. Formulario OM-4. 61
Tabla 6. Formulario OM-5. 62
Tabla 7. Formulario TM-1. 64
Tabla 8. Formulario TM-2. 65
Tabla 9. Formulario AM-1. 66
Tabla 10. Formulario OTA-1. 67
Tabla 11. Formulario KM-1. 69
Tabla 12. Formulario CM-1. 73
Tabla 13. Formulario CM-2. 73
Tabla 14. Formulario DM-1. 78
Tabla 15. Formulario DM-2. 78
Tabla 16. Formulario DM-3. 79
Tabla 17. Formulario DM-4. 80

INTRODUCCIÓN

La capacidad del razonamiento humano se ha convertido en un objeto de estudio en el cual científicos de distintas épocas se han embarcado, con el fin de poder crear y dotar a una máquina que simule de forma similar el raciocinio de las personas frente a las situaciones de la vida cotidiana en las que se ve envuelto un proceso de toma de decisiones. Estos esfuerzos han dado paso al desarrollo de herramientas destinadas a la creación de sistemas inteligentes o sistemas basados en el conocimiento (SBC).

Un SBC trabaja utilizando una base de hechos que son suministrados por un experto en un área del conocimiento específica y representados por los ingenieros del conocimiento en un lenguaje comprensible por las computadoras.

El desarrollo de un SBC comparte características con la Ingeniería de Software (IS) para la construcción de un sistema, no obstante, las metodologías utilizadas en IS no contemplan un tratamiento especial al recurso principal de un SBC: el conocimiento, el cual es variable, aun dentro de un solo dominio; por esto, se han diseñado metodologías orientadas al desarrollo de sistemas inteligentes o sistemas expertos, entre las que se encuentra CommonKADS.

La metodología CommonKADS ofrece un conjunto de formularios genéricos que deben completarse a medida que avanza el estudio y desarrollo del sistema, debido a que en las etapas finales toda la información recolectada se integra en un solo documento.

El presente documento contiene los conceptos básicos de un SBC, ejemplos de herramientas que se utilizan en el desarrollo de este tipo de sistemas, las fases de la metodología CommonKADS y técnicas de representación del conocimiento.

1. CONCEPTOS, CARACTERÍSTICAS Y ESTRUCTURA BÁSICA DE LOS SBC

Los avances tecnológicos que han surgido con el pasar de las décadas, han dejado marcadas las diferencias entre los sistemas creados dentro del margen de la inteligencia artificial, en los que se consideran aspectos como el tipo de procesamiento que es aplicado sobre los datos con los que se alimentan dichos sistemas, tal es el caso de los sistemas basados en el conocimiento (SBC). Este capítulo se enfoca en definir los conceptos fundamentales, características y estructuras que poseen los SBC.

1.1. Definición de los SBC

Los sistemas basados en el conocimiento (SBC) son una clase de sistemas capaces de albergar una representación del conocimiento de un área de aplicación en específico, estando en la posibilidad de brindar soluciones óptimas frente a una problemática dada, mediante el uso de métodos de razonamiento elaborados para sacar el mayor provecho del conocimiento almacenado y poder transmitirlo de manera similar a la que lo haría un experto (Garcia & Rodríguez, 2013). A menudo se suele referir a los SBC como sistemas expertos.

Este tipo de sistemas requiere el apoyo de equipo computacional en el que será capturado el conocimiento del experto y el conjunto de reglas que se utilizarán para la generación del nuevo conocimiento.

1.1.1. Características de los SBC

Un SBC o sistema experto debe contar con una serie de características particulares (Amatriain, 2015) que lo hacen diferenciarse de otros tipos de software tradicionales, entre ellas se encuentra:

- Se especializa en un área de conocimiento específica.
- Resuelve problemas con un grado de dificultad elevado, propios de un dominio determinado.
- Posee la capacidad de manejar grandes volúmenes de información.
- Utilizan razonamiento simbólico.

- Aplica nuevas estructuras y modos de organización del conocimiento.
- Aplica el conocimiento que posee para realizar inferencias a partir de nuevos datos.
- Manejan incertidumbre.
- No usan procedimientos algorítmicos.
- Procuran obtener las soluciones óptimas.
- Emplean datos mayormente cualitativos.
- Pueden comunicarse con expertos para nutrir su base de conocimientos.
- Justifican los resultados obtenidos en caso de ser solicitado por el experto o usuario.
- Debe estar en capacidad de manipular, modificar, actualizar y ampliar datos.

A pesar de que las características mencionadas son las ideales, obtenerlas todas dentro de un SBC es complicado y pocas veces alcanzable para los desarrolladores, dadas las condiciones de tiempos de entrega y recursos.

1.1.2. Tipos de SBC

Los SBC varían en la manera en que son diseñados y construidos, estas diferencias radican con mayor incidencia en el método escogido para almacenar, procesar e inferir el nuevo conocimiento.

De esta forma, los SBC se clasifican considerando su aplicación o técnica de razonamiento (Badaró, Ibañez, & Agüero, 2013). Dentro del marco de modo de representación y manipulación del conocimiento para la generación de respuestas o soluciones, se tienen los siguientes tipos de SBC:

a) **Basados en reglas:** Trabaja con hechos y reglas, siendo características estructuras de control como IF-THEN, etiquetados, entre otros; mediante los que se van comparando los resultados obtenidos dadas determinadas condiciones iniciales.
b) **Basados en casos**: Su mayor distinción está en el uso de la experiencia obtenida para la resolución de un problema para solventar situaciones

futuras. Este tipo de SBC hace una analogía con la forma en la que el ser humano tiende a razonar en su diario vivir.

c) **Basados en redes bayesianas**: Emplea probabilidades para determinar las relaciones entre un conjunto de variables dentro de un modelo de grafo acíclico dirigido.

d) **Basados en lógica difusa**: La incertidumbre es su característica principal; trata de simular el razonamiento humano, que tiende a tener más opciones que un simple Sí o No. Esta incertidumbre es aplicada mediante conjuntos difusos, que causan que el sistema trabaje de forma menos precisa, pero con una mayor lógica "humana".

1.2. Componentes de los SBC

Los SBC poseen una estructura básica en la que se distinguen una serie de componentes cuyas relaciones se pueden apreciar en la figura 1.

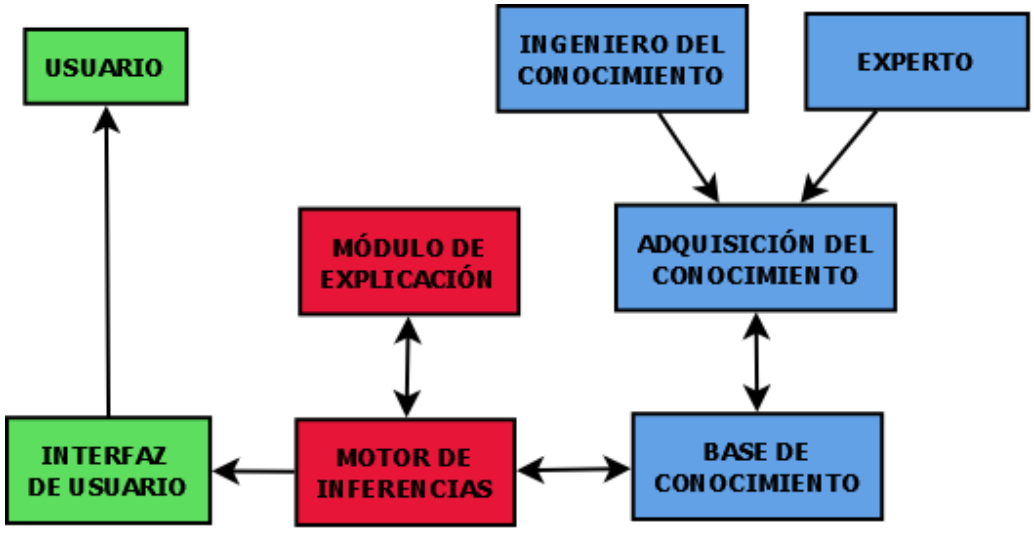


Figura 1. Estructura de un SBC.

A continuación, se procede a definir en qué consiste cada uno de los elementos que forman la arquitectura de un SBC.

1.2.1. Base del conocimiento

En ella se almacena el conocimiento de los hechos y la heurística, factores necesarios para comprender, formular y ofrecer una solución a los problemas propuestos. Se alimenta mediante el módulo de adquisición de conocimiento.

1.2.1.1. Tipos de conocimiento

Se distinguen dos tipos de conocimiento:

a) **Factual**: Es aquél que es ampliamente compartido y que puede encontrarse en revistas o libros, corresponde al dominio de la aplicación.
b) **Heurístico:** Se refiere al conocimiento de las buenas prácticas, buen juicio y el razonamiento plausible en el campo. Este tipo de conocimiento no es fácil de representar, dado su grado de subjetividad e incertidumbre.

1.2.1.2. Estrategias para representar el conocimiento

Al momento de seleccionar una estrategia para la representación del conocimiento que será empleado por el SBC, es importante tener en claro algunas características como la capacidad de generalizar, comprensibilidad, facilidad de modificación e incremento, entre otras; pues esto determinará la complejidad que deben tener los métodos diseñados para cumplir la labor de inferir las soluciones (Carlos Soto, 2002).

Entre las estrategias para representar el conocimiento están:

- **Autómatas finitos**: Descompone el conocimiento en fragmentos estáticos, es decir, difíciles de alterar o cambiar.
- **Cálculos de predicados**: Emplea lógica de proposiciones.
- **Reglas de producción**: Hace uso de condiciones y comparativas.
- **Redes semánticas**: Utiliza grafos en donde los nodos son conceptos y los arcos, relaciones.
- **Objetos estructurados**: Combina las anteriores para obtener una representación más acertada.

1.2.2. Motor de inferencias

Actúa como el cerebro del sistema, en el que se emplean estructuras de control y una metodología para efectuar el razonamiento. Esta tarea toma como base respuestas previas dadas por los usuarios y las correspondientes reglas de activación. Su misión es encontrar un camino dentro de un conjunto de reglas para llegar a una conclusión.

Este camino puede ser encontrado utilizando dos tipos de estrategias:

a) **Encadenamiento hacia adelante**: Aplicable cuando se tiene un número de resultados muy amplio por lo que es necesario construir la solución, acción realizada evaluando los hechos conocidos contra las reglas aplicables en la base de conocimiento (Harmon & King, 1988).
b) **Encadenamiento hacia atrás**: Mayormente empleado cuando la cantidad de resultados posibles es conocido y bastante reducido; en donde el proceso realizado conlleva determinar qué reglas pueden llevar al objetivo o solución que se espera (Harmon & King, 1988), es decir, que se van desglosando todas las condiciones que son necesarias para que ocurra ese objetivo a medida que se van encontrando reglas aplicables (Pino Díez, Gómez Gómez, & de Abajo Martínez, 2001).

1.2.3. Módulo de adquisición del conocimiento

El módulo de adquisición de conocimiento es el encargado de capturar el conocimiento, ya sea de los expertos o el que está documentado en las fuentes bibliográficas que se emplean para la construcción del sistema experto. Su misión principal es ayudar a construir las bases del conocimiento permitiendo su posterior ampliación o actualización.

Entre las técnicas empleadas para la adquisición del conocimiento se encuentran los análisis de protocolo, entrevistas y observación.

1.2.4. Módulo de explicación

Es una unidad creada con el fin de poder ofrecer información adicional acerca de los resultados arrojados por el sistema. Las preguntas o solicitudes de las razones de los resultados pueden ser realizadas por los usuarios. Esta función contribuye con una de las misiones de los SBC que es la transmisión del conocimiento como si tratase de un tutor.

1.2.5. Interfaz de usuario

Como toda herramienta de software que se ofrecerá a un público determinado, es necesario que cuente con una interfaz lo suficientemente agradable y comprensible por los mismos, en los que se proporcione menús y ventanas de diálogos que faciliten su uso. En la interfaz se muestra el contenido de las reglas almacenadas en la base de conocimiento de forma entendible para cualquier persona.

1.3. Desarrollo de un SBC

El recurso fundamental para iniciar un proyecto en el que se busca construir un SBC es el conocimiento en sí, debido a que este es la razón de existir del sistema. El motivo principal por el que se implementa un SBC es la necesidad de transmitir y ofrecer acceso permanente al conocimiento de un experto, a cualquier persona indistintamente de su nivel de educación, de manera que este pueda ser aprovechado en su totalidad.

1.3.1. Etapas de desarrollo de un SBC

El desarrollo de un SBC, como cualquier otro tipo de programa convencional, tiene una serie de etapas o fases para su construcción desde cero. Distinguimos cinco etapas, cada una con sus características:

1.3.1.1. Identificación del problema

Siendo el primer paso en que se involucra un alto nivel de análisis y estudios de factibilidad, se debe tener un especialista en el área en el que se quiere aplicar el SBC y un ingeniero del conocimiento. El especialista es quien posee el

conocimiento que se quiere capturar en el sistema; mientras que el ingeniero del conocimiento es quien está en la capacidad de extraer los conceptos más relevantes de ese dominio, creando objetos para su representación e identificando sus relaciones (Russell & Norvig, 2004). El ingeniero del conocimiento debe tener en consideración lo siguiente (Garcia & Rodríguez, 2013):

a) Tipo de conocimiento.
b) La forma en que ese conocimiento se aplica en la resolución de problemáticas.
c) La reacción del experto frente a las problemáticas.

Ambos deben discutir y definir aspectos como el alcance y los recursos que se necesitarán para el desarrollo del sistema, es decir, un análisis en el que se demuestre el beneficio obtenido a cambio de la inversión que se va a realizar para la implementación del SBC.

Un punto que también debe ser analizado es la posible presencia de espacios que den lugar a problemas que sobrepasen la capacidad del sistema, sugiriendo de forma conceptual planes de contingencia que funcionen para reducir la gravedad de esos problemas.

1.3.1.2. Modo de desarrollo

Un recurso indispensable para poder adentrarse en esta fase, son los requerimientos del SBC, discutidos en la fase anterior. En base a los requerimientos, el ingeniero del conocimiento puede proceder a la selección de las herramientas necesarias para la obtención de las características deseadas en el sistema.

Actualmente existe una gama de lenguajes que permiten realizar un desarrollo desde cero, y a la vez, se tiene a disposición otras, conocidas como Shell, que además de facilitar el proceso, incluyen estructuras prediseñadas, a las cuales se les aplican las modificaciones que sean necesarias a través de una interfaz gráfica en la que el ingeniero del conocimiento o el experto puede registrar la experiencia y codificar el conocimiento.

1.3.1.2.1. Elección de lenguajes, herramientas y conchas

Una herramienta para el desarrollo de SBC no es más que un software que contiene una serie de instrucciones y facilidades creadas con el propósito de ser utilizadas para este fin, ya que incluyen funciones para el aprendizaje, generación de explicaciones e inferencias (Sajja & Akerkar, 2010); no obstante, también pueden emplearse lenguajes de programación de propósito general.

Entre los lenguajes creados para la programación de SBC se tienen:

- **PROLOG**: Creado en la década de 1970. Es un lenguaje de programación de sintaxis y semántica sencilla, en la que se deben declarar los hechos, reglas y preguntas para inferir el conocimiento (Clocksin & Mellish, 2012). Hace uso de estructuras de árboles y un enfoque de encadenamiento hacia atrás.
- **LISP**: Lenguaje basado en listas (Noyes, 1992), debido a esto, todos los datos se manejan en un solo tipo de estructura, que suele ser manejada por funciones matemáticas. Una versión de este lenguaje es el conocido como CLIPS.

De igual forma, se tienen Shells o conchas como:

- **Java Expert System Shell (JESS):** Soporta el desarrollo de sistemas basados en reglas. Posee similitudes en la sintaxis de CLIPS.
- **Jena:** Framework utilizado para la inferencia del conocimiento basada en el uso de ontologías. Brinda una API ontológica y su propio motor de búsqueda.
- **JEOps**: Trabaja con reglas de producción y encadenamiento hacia adelante, por lo tanto, sigue el principio de la programación declarativa.

1.3.1.3. Desarrollo de un prototipo

Se lanzan las primeras versiones del sistema, en las que se realiza un proceso de pruebas y rectificación sobre la base de conocimiento que ha sido construida, tal cual se realizaría para un desarrollo de software convencional, en el que se ejecutan validaciones y verificaciones. Es necesario contemplar:

1. Los conceptos claves para producir la solución.
2. Decidir el nivel de conocimiento del sistema, es decir, su granularidad.
3. Se deben realizar reuniones frecuentes entre el experto y el ingeniero del conocimiento, con el propósito de extraer lo más posible el conocimiento del dominio.
4. Se verifica que la representación del conocimiento cumpla con el propósito buscado, de lo contrario debe ser modificado.

Terminado este proceso, se obtiene la versión final para entrega a los interesados del proyecto, ya con todos los ajustes a los problemas que hayan sido detectados durante las pruebas.

1.3.1.4. Planificación de un sistema a gran escala

Es común que, dado el éxito de la implementación de un sistema pequeño, se busque la manera de incrementar su tamaño y potencia, este motivo es por el cual precisamente la elección adecuada de las herramientas, así como una correcta estructuración del conocimiento en las fases iniciales del proyecto, determinan la posibilidad y flexibilidad del sistema ante cambios o modificaciones futuras.

En esta fase es frecuente el uso de diagramas de Gantt, PERT o CPM; para garantizar un uso adecuado de los recursos en el mejor tiempo posible.

1.3.1.5. Implementación, mantenimiento y evolución

En esta fase el sistema es colocado en un entorno en donde entrará en contacto directo con sus usuarios finales, quienes no necesariamente fueron los que impulsaron el desarrollo del SBC.

El mantenimiento y evolución implica una verificación periódica del conocimiento aplicado en el sistema, y realizar las debidas actualizaciones y adiciones de nuevo conocimiento en caso de ser necesario.

1.4. Aplicaciones de un SBC

Un SBC es creado con el propósito de ser aplicado en un área o tarea determinada, de modo que pueda servir como guía o consulta dependiendo del

nivel de conocimiento de su usuario. Teniendo esto en consideración, se puede subdividir a los SBC de acuerdo con las funciones que realicen:

- **Clasificación:** Diseñados para Identificar un objeto en base a las características que presenten.
- **Sistemas de diagnóstico**: Dados una serie de datos observables, proporcionados por el usuario, infiere qué problema, malfuncionamiento o enfermedad se está presentando.
- **Monitoreo**: Prescribe el comportamiento de un sistema, mediante la comparación continua de los datos generados por el mismo en el tiempo.
- **Control de procesos**: Mediante un seguimiento se verifica la correcta ejecución de las tareas o procesos.
- **Diseño**: Orientado a generar un resultado basado en las especificaciones indicadas por el usuario.
- **Planificación**: Desarrollar o modificar un plan de acción tomando en cuenta determinados parámetros de eficiencia.
- **Generación de opciones**: Brinda soluciones o alternativas a un problema, cada una con un grado menor o mayor de eficacia.

1.4.1. Razones para utilizar un SBC

Luego de haber visto los conceptos fundamentales y el ciclo de vida de los SBC, hay que destacar las situaciones en las que es realmente válida la implementación de un sistema de este tipo:

a) No hay suficientes expertos humanos en un área del conocimiento.
b) Cuando el conocimiento debe ponerse a disposición constante.
c) Situaciones en los que la subjetividad humana, pueda acarrear soluciones equivocadas.
d) El volumen de datos a procesar es muy elevado para obtener un resultado.
e) Cuando el conocimiento de más de un experto debe ser almacenado en una sola plataforma.

1.4.2. Ventajas y Limitaciones

Entre las ventajas más sobresalientes de la aplicación de un SBC para la solución de problemas están:

a) Mayor rapidez en la obtención de respuestas.
b) Reducción del error humano.
c) Disminuyen la inversión de salarios de recursos humanos.
d) Colocan a disposición de cualquiera el conocimiento que almacenan.
e) Pueden estar en servicio en cualquier momento.
f) Capacidad para trabajar con información parcial como un experto humano.
g) Aprende de su experiencia previa.

Por otra parte, algunas de las limitaciones detectadas de forma general para el uso de un SBC, son las siguientes:

a) Carecen de sentido común.
b) No pueden mantener una conversación en lenguaje natural.
c) Para que identifique errores es necesario intervenir a nivel de programación.
d) Requiere que se ingrese conocimiento estructurado.

2. HERRAMIENTAS PARA EL DESARROLLO DE SBC

Para el desarrollo de un SBC se puede hacer uso de cualquier lenguaje de programación de uso genérico, sin embargo, el proceso puede tornarse más lento y extenso dado que se requiere diseñar y codificar el sistema desde cero; no obstante, existe una gama de software creados con el fin de crear sistemas de este tipo, que incluyen sus propias estructuras para la representación del conocimiento, así como sus propios motores de búsqueda e inferencias. Este capitulo se dedica a repasar el uso básico de algunas de las herramientas para el desarrollo de SBC que fueron vistas en cursos previos como Inteligencia Artificial y Herramientas Aplicadas a la Inteligencia Artificial.

2.1. Expert System Builder

Mejor conocido como ES-Builder, es una herramienta gratuita que permite desarrollar las bases de las habilidades para la construcción de un SBC, pues ofrece una interfaz en la que se puede observar cómo es ordenado el conocimiento siguiendo una estructura de árbol de decisión.

Desarrollado por McGoo Software, con fines educativos, en un principio contaba con dos versiones: la versión de escritorio para Windows y la versión web; sin embargo, actualmente solo se encuentra disponible la versión web.

2.1.1. Nodos

ES-Builder, al trabajar con árboles de decisión, utiliza nodos, específicamente tres tipos de nodos que responden a una jerarquía dentro del árbol: *Attributes, Values* y *Conclusion*. Estos tres tipos de nodos son albergados en su totalidad dentro de un nodo raíz o universo, que, desde una perspectiva de SBC, vendría siendo el dominio de la aplicación, por lo tanto, el nodo raíz es el primer nivel del árbol.

En los siguientes puntos se procede a definir las características y roles que cumplen cada tipo de nodos.

2.1.1.1. Attributes

El nodo *Attribute* o atributo, se encuentra en el segundo nivel del árbol. Este nodo está representado mediante una letra 'A' en color azul.

Un atributo corresponde a una categoría, característica o rasgo general, que son propias para llegar a una conclusión. Ejemplos de atributos: color, tamaño, forma, etc.

2.1.1.2. Values

En el caso del nodo *Value* o valor, se encuentra justo en el siguiente nivel que un nodo de atributo. Es representado con la letra 'V' en tonalidad rojo vino. Este nodo presenta dos comportamientos dependiendo del nivel de complejidad o especificidad del dominio que se está implementando.

a) **Sub-categoría**: Este caso se da cuando dentro de un atributo, pueden darse múltiples valores, por ejemplo, el tamaño de un objeto. En estas circunstancias, el tamaño podría ser grande, mediano o pequeño.
b) **Condición de una sola conclusión**: Aquí el nodo valor no contiene otros atributos, es decir, no presenta otras ramas, por ende, solo da paso a la conclusión. Es importante señalar que solo puede existir una conclusión dentro de un nodo valor.

2.1.1.3. Conclusion

El nodo *Conclusion* o conclusión, por su parte, cumple el papel de hoja en el árbol, ya que una conclusión es el nodo final de una rama. Su representación es una letra 'C' en color verde.

2.1.2. Ejemplo: Razas de perros

Para efectos ilustrativos del proceso de creación de un sistema en ES-Builder, se procede a construir un ejemplo basado en un caso sencillo para la identificación de razas de perros.

Lo primero es crear una cuenta e iniciar sesión en la misma. De esta manera se accede a la pantalla de inicio de la herramienta. En esta pantalla se da clic en el icono de que muestra una regla y un lápiz formando una 'X', que corresponde al

comando **“Create a new Project…”**; esto hará que se muestre el formulario para la creación de un nuevo proyecto (Figura 2).

En el formulario se ingresa la información básica del proyecto: nombre, la especificación del universo o dominio del sistema, así como datos adicionales que serán omitidos en este ejemplo.

El nombre del sistema será **“Razas de perros”** y el universo será **“Clasificación de algunas razas de perros”**. Para finalizar el proceso de creación se da clic en el botón **“Save the New Project”**.

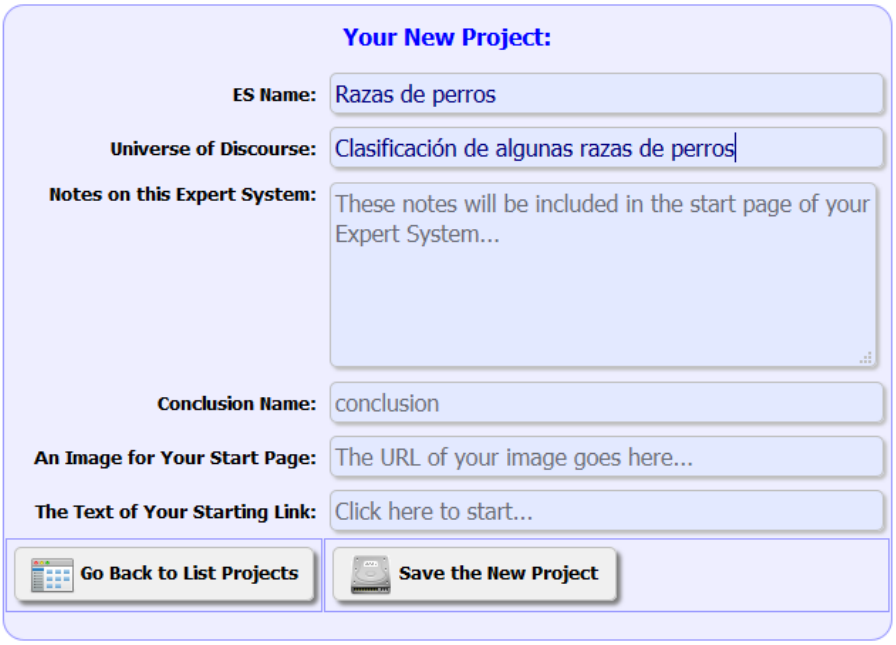


Figura 2. Formulario para la creación de un nuevo proyecto en ES-Builder Web.

La acción de guardar el proyecto, tiene por efecto retornar a la pantalla de inicio, en la que se aprecia el proyecto creado en la lista de **“Your Projects”** (Figura 3). Para crear los nodos del sistema, se da clic en la opción **“Edit”,** cuyo icono es una hoja de papel y un lápiz.

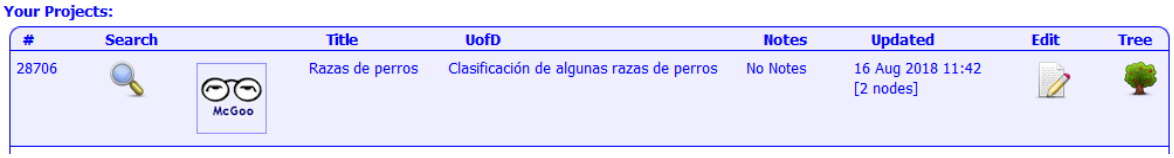


Figura 3. Actualización de la lista de proyectos.

Dentro de la nueva ventana, del lado derecho del observador, hay una columna con una serie de botones, se debe dar clic en el botón **"Edit Tree"** (Figura 4) para iniciar el proceso de creación del árbol de decisiones del sistema. El árbol por defecto solo muestra un atributo y un valor por definir (Figura 4).

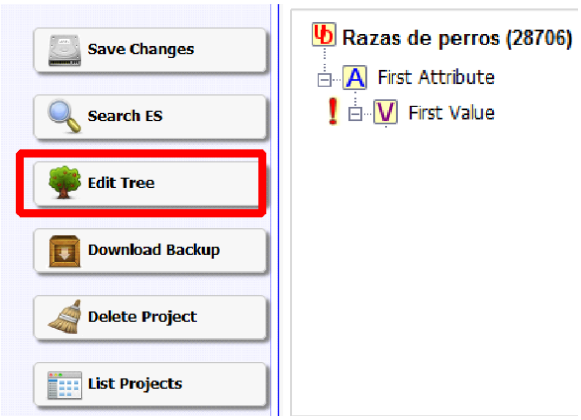


Figura 4. Opción Edit Tree (izquierda) y vista del árbol por defecto del sistema (derecha).

El primer nodo del ejemplo es **"Tamaño"**, por lo tanto, para cambiar el nombre del primer atributo A, se hace doble clic sobre el mismo y se ingresa el nuevo nombre. De esta misma forma se cambia el nombre del primer valor a **"Grande"**.

Para añadir un nuevo atributo dentro del nodo **"Grande"**, basta con hacer clic derecho sobre el mismo, de esta manera se despliega un menú que muestra diferentes acciones a realizar en el nodo (Figura 5).

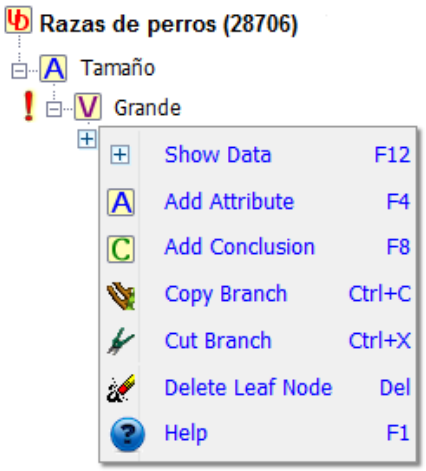


Figura 5. Menú de opciones dentro de un nodo valor.

Como se puede ver en la figura 5, dentro de un nodo V es posible añadir un nodo A o un nodo C. En este caso se añadirá un nodo A, que debe ser nombrado **"Pelaje"**. Dentro de **"Pelaje"** se añaden tres nodos V. El primero tendrá por nombre **"Pelo largo",** el segundo **"Pelo corto con manchas"** y el tercero **"Pelo corto sin manchas"**. El comportamiento de estos nodos V corresponde al de subcategoría, mencionado en el punto **a** de la sección 2.1.1.2.

Para **"Pelo largo"** se debe añadir un nodo C **"Golden Retriever"**. Un nodo C representa una conclusión y por tanto el nodo final u hoja de una rama.

Por otra parte, dentro del nodo **"Pelo corto con manchas"** y **"Pelo corto sin manchas"**; dentro de lo cuales irán los nodos C: **"Dálmata"** y **"Labrador"**, respectivamente. Así culmina el nodo **"Grande"** del árbol de decisión (Figura 6).

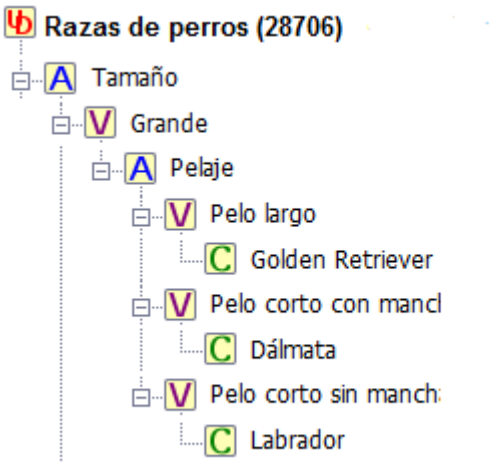


Figura 6. Nodo A, Tamaño, terminado.

La segunda rama, dentro del nodo tamaño, será **“Pequeño”**. Dentro de este nodo se creará un nodo A **“Orejas”** y a su vez, este contendrá dos nodos V: **“Erguidas”** y **“Caídas”**. Las conclusiones de estos dos nodos son: **“Terrier”** y **“Beagle”**. En la figura 7 se muestra el árbol completo.

Es recomendable que a medida que se avanza en la construcción del árbol, se guarden los cambios realizados, dado que la herramienta no tiene una función de autoguardado. Para ejecutar el sistema, sólo se debe dar clic en el icono de la lupa.

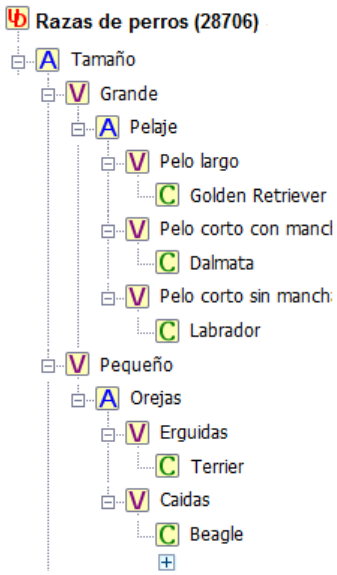


Figura 7. Árbol terminado.

2.2. ExSys Corvid

Exsys Corvid es una poderosa herramienta, de propósito general, con un ambiente de desarrollo intuitivo, en el que los expertos pueden describir el conocimiento mediante un conjunto de reglas en el idioma inglés y un poco de algebra (Exsys Corvid, 2011). Esta herramienta ha sido diseñada con el fin de no requerir conocimientos de programación para su utilización en el proceso de creación de sistemas expertos.

Al igual que el sistema anterior utiliza estructuras de árboles, además de permitir el uso de estrategias de razonamiento de encadenamiento hacia adelante y hacia atrás. Otra ventaja que se debe destacar es su entorno para el diseño, creación y edición de interfaces de usuarios, dando paso a una aplicación más interactiva que solo desplegar preguntas en una pantalla.

Para la utilización de Exsys Corvid en la creación de sistemas inteligentes, se debe seguir una secuencia general en la que primero se definen las variables a emplear por el sistema, luego los bloques lógicos que contienen las reglas que serán evaluadas y bloques de comandos que determinan ciertas acciones en el sistema.

2.2.1. Variables

Las variables dentro de Corvid poseen similitud con las variables de cualquier otro lenguaje de programación, por lo tanto, son elementos nombrados por el programador y están asociados a un tipo de valor. Las variables son un elemento fundamental para el diseño de un sistema, pues pueden adoptar el valor que el usuario indique, ser probadas y modificadas por las reglas que rigen al sistema inteligente, debido a esto, se deben seguir ciertas pautas al momento de su definición:

1) **Name (Nombre):** Debe ser único y no debe contener caracteres especiales. No existe límite de longitud; no obstante, se recomienda que sea lo más específico y corto posible pero que refleje su papel dentro del sistema.
2) **Prompt (Mensaje, aviso):** Es el enunciado o pregunta que se va a mostrar al usuario. Debe estar relacionado con el tipo de valor que se espera almacenar en la variable.

Tanto el Name como el Prompt, son componentes de las variables. Es permitido asignar más de un Prompt para una variable, esto representa una ventaja si el sistema debe ser ejecutado en varios idiomas.

Corvid cuenta con un total de siete tipos de variables, cada una con funcionalidades y capacidades determinadas. En la tabla 1 se detallan los tipos de variables que maneja Exsys Corvid.

Tabla 1. Tipos de variables en Exsys Corvid.

Tipo de variable	Características
Static list	✔ Contiene una cantidad definida de posibles valores para una variable. ✔ Empleada en casos de selección múltiple.

Dynamic list	✔ Lista de posibles valores que cambian durante la ejecución. ✔ Es utilizada cuando se desconocen los valores de una variable durante el proceso de construcción del sistema.
Numeric	✔ Se usa para variables que deben almacenar valores numéricos en general. ✔ Su valor puede ser asignado por el usuario, calculado por las reglas, obtenido de recursos externos, etc.
String	✔ Indicado para almacenar texto. ✔ Sus usos comunes radican como identificadores de recursos externos y texto que será colocado en reportes.
Date	✔ Permite almacenar información sobre fechas, desde el nombre del día hasta el tiempo en milisegundos. ✔ Generalmente se utilizan en comparaciones de tiempo o son calculadas mediante funciones de Corvid.
Collection / Report	✔ Consiste en una lista de cadenas de texto, cualquier tipo de valor en formato de cadena. ✔ Cuenta con un número de funciones para añadir, remover, ordenar y probar los elementos que contiene. ✔ Como su nombre indica, es usado para la generación de reportes.

Confidence	✔ Consiste en un valor de probabilidad o grado de certeza de una recomendación. ✔ Se emplea mayoritariamente en sistemas que trabajan con lógica difusa, en los que existen más de una solución o respuesta.

2.2.2. Creación de Bloques Lógicos

Las reglas en Exsys Corvid son agrupadas dentro de los llamados bloques lógicos. Estos bloques pueden contener desde una simple regla hasta varias estructuras de árboles de reglas con la única condición de que estas se encuentren relacionadas entre sí para la resolución de un aspecto en la toma de decisiones.

La manera en que se organizan las reglas dentro de los bloques lógicos sigue la estructura del IF-THEN, es decir, IF alguna o varias condiciones son ciertas THEN alguna(s) otra(s) condición(es) pueden ser ciertas, como se muestra en la figura 8.

Corvid ofrece una ventana en la que puede apreciar la manera en la que se está estructurando el conocimiento. Esto permite que los bloques de reglas sean construidos siguiendo distintas lógicas, ya que cada experto tiene una forma de resolver una problemática.

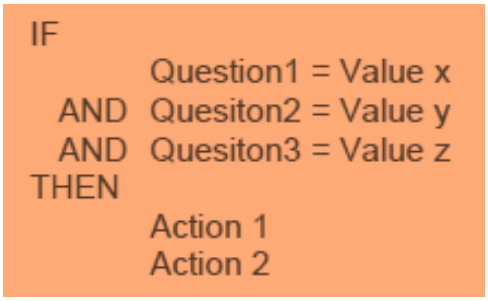


Figura 8. Estructura de las reglas en Exsys Corvid.

La escritura de las reglas debe ser lo más breve posible, a modo de variable; de igual forma el valor o conclusión de la regla debe ser corto, pero lo suficientemente representativo.

2.2.3. Bloques de Comandos

Los bloques lógicos determinan la manera cómo realizar las cosas, mientras que los bloques de comandos le indican al sistema qué hacer. Un bloque de comandos le dice al motor de inferencia de Corvid la manera en que se deben aplicar las reglas. Esta cualidad de Corvid facilita la forma de construir y dar mantenimiento a los sistemas. Es imprescindible que exista al menos un bloque de comandos en cualquier sistema creado usando Corvid, de lo contrario el programa no podrá ser ejecutado.

Un bloque de comando realiza cualquier tipo de acción en el sistema, sin afectar la lógica de este.

Entre los usos comunes para los bloques de comando están los siguientes:

- ✔ Determinar el enfoque de encadenamiento, hacia adelante o hacia atrás.
- ✔ Controlar el orden en que se ejecutan los bloques lógicos.
- ✔ Obtener datos de recursos externos al sistema.
- ✔ Transmitir los resultados y recomendaciones a otros programas.
- ✔ Aplicar estructuras de repetición como FOR y WHILE.
- ✔ Mostrar mensajes y ventanas de ayuda al usuario.
- ✔ Mostrar reportes.
- ✔ Enviar correos electrónicos.

Dadas las funciones que un bloque de comando puede realizar, la ventana en la que son añadidos posee una serie de pestañas de las cuales se usan con mayor frecuencia las de Variables, Bloques y Resultados.

a) **Pestaña de Variables:** Se usa para definir directamente un valor a una variable o cuando se debe derivar su valor mediante encadenamiento hacia atrás.

b) **Pestaña de Bloques:** Proporciona los comandos requeridos para correr un sistema que utiliza encadenamiento hacia adelante, especificando el orden en que deben verificarse las reglas.
c) **Pestaña de Resultados:** Tal como su nombre lo indica, tiene las opciones para el despliegue de resultados, ya sea de forma simple, es decir, texto sin formato; hasta establecer un formato de salida en el que no aparecen todas las variables involucradas en el proceso de generación de las recomendaciones.

Luego de construido el sistema, Corvid realiza la ejecución del programa desde la barra de comandos. Para esto se requiere contar con las últimas actualizaciones de Java Runtime. Corvid emplea Java para hacer sus aplicaciones compatibles con entornos web.

Para explorar el uso básico de las opciones que brinda Corvid, en el siguiente punto se desarrollará un ejemplo sobre los requisitos para la solicitud de un préstamo en una entidad bancaria.

2.2.4. Ejemplo: Solicitud de Préstamo

Con el objetivo de aclarar los aspectos teóricos expuestos en los puntos anteriores, en esta sección se desarrolla un ejemplo utilizando requisitos básicos para la solicitud de un préstamo, en el cual el sistema determinará si el cliente cumple o no con los requisitos para su aprobación.

2.2.4.1. Paso 1: Crear el proyecto

Para la creación del proyecto, se debe abrir el programa Exsys Corvid y en el menú seleccionar **FILE** -> **New** (Ver 1 en figura 9). Luego se asigna un nombre al proyecto, en este caso **Préstamo** (Ver 2 en figura 9) y se da clic en la opción **Create New Project** (Ver 3 en figura 9). Con esto ya está listo el archivo para trabajar en él.

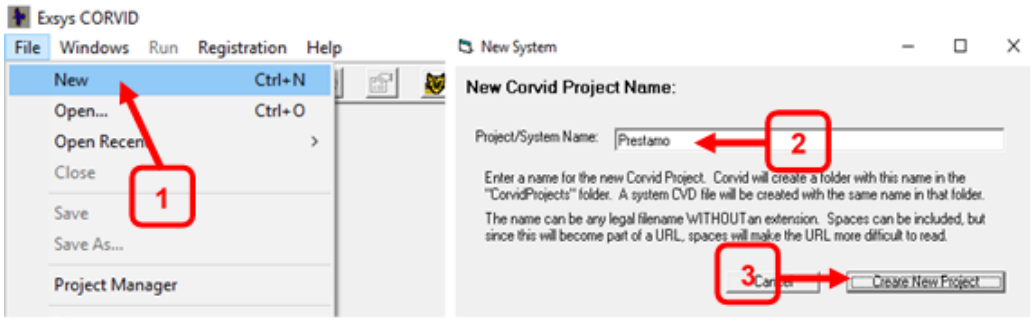


Figura 9. Pasos para la creación de un nuevo proyecto en Corvid.

2.2.4.2. Paso 2: Declarar las variables y asignar sus valores

Los elementos principales dentro de un sistema desarrollado en Corvid son las variables, por lo tanto, es necesario crear algunas para que el sistema pueda funcionar; para esto una vez que se crea el nuevo proyecto, la ventana de **Variables** es desplegada. En caso contrario, se abre haciendo clic en el icono de variable (Ver 1 en figura 10).

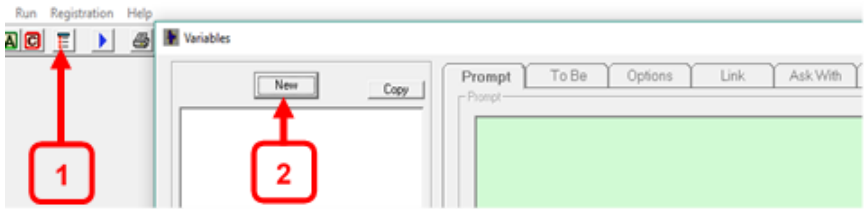


Figura 10. Ventana para la creación de variables.

Seguido a esto, dentro de la ventana de variables, se selecciona el botón **New** (Ver 2 en figura 10).

La primera variable que crear será **"Ingreso_Cliente"** que tendrá dos posibles valores **"menos de B/.600.00"** y **"al menos B/.600.00"**.

Al hacer clic en el botón **New** se despliega la ventana de **New Variable**, en la cual se ingresará "Ingreso_Cliente" en el campo **Name** (ver 1 en figura 11) y se debe verificar que la opción de **Static List** esté seleccionada (ver 2 en figura 11). Para finalizar la creación se da clic en **OK** (ver 3 en figura 11)**.**

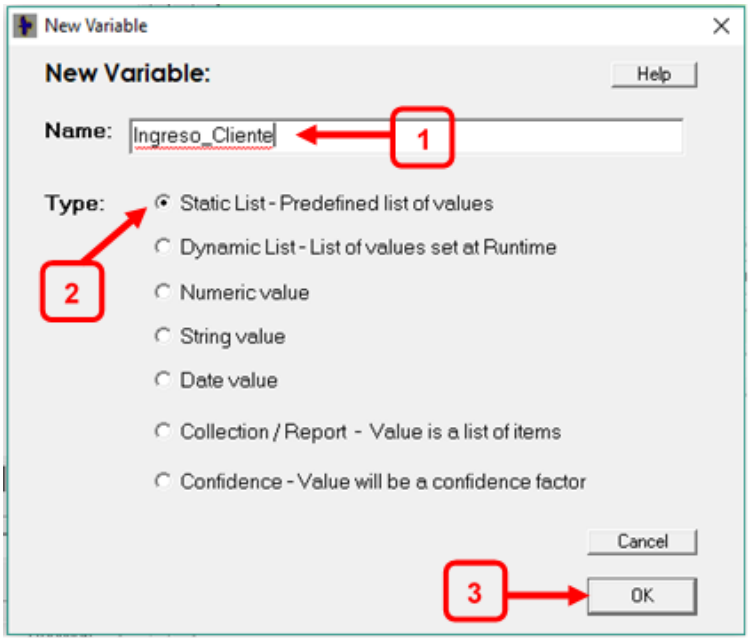


Figura 11. Creación variable Ingreso_Cliente.

Para añadir los valores que puede tomar la variable, se debe seleccionar **"Ingreso_Cliente"** de manera que quede resaltada en azul (ver 1 en figura 12), pero antes, el nuevo **Prompt** para esta variable es **"El ingreso del cliente es"** (ver 2 en figura 12).

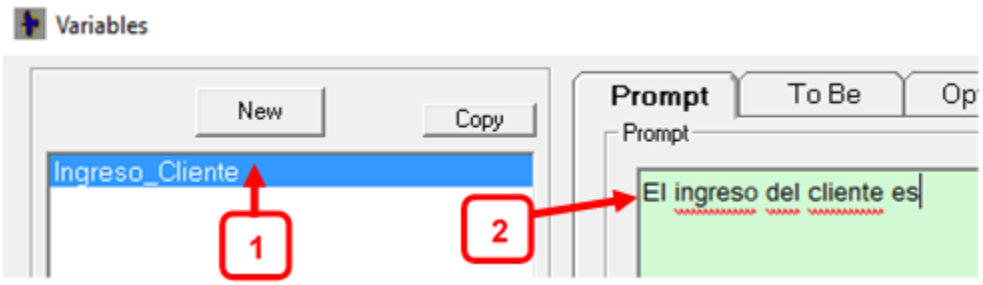


Figura 12. Modificación del Prompt de la variable Ingreso_Cliente.

El primer valor que añadir será **"menos de B/.600.00"**, este texto debe ser ingresado en el campo **Value** de la pestaña **Static List** (ver 1 en figura 13), ubicada en el cuadrante inferior derecho de la ventana **Variables**. Acto seguido se presiona **Enter** o se hace clic en la opción **Add to List** (ver 2 en figura 13). Este mismo procedimiento debe ser realizado con el valor **"al menos B/. 600.00"** (ver 3 y 4 en figura 13).

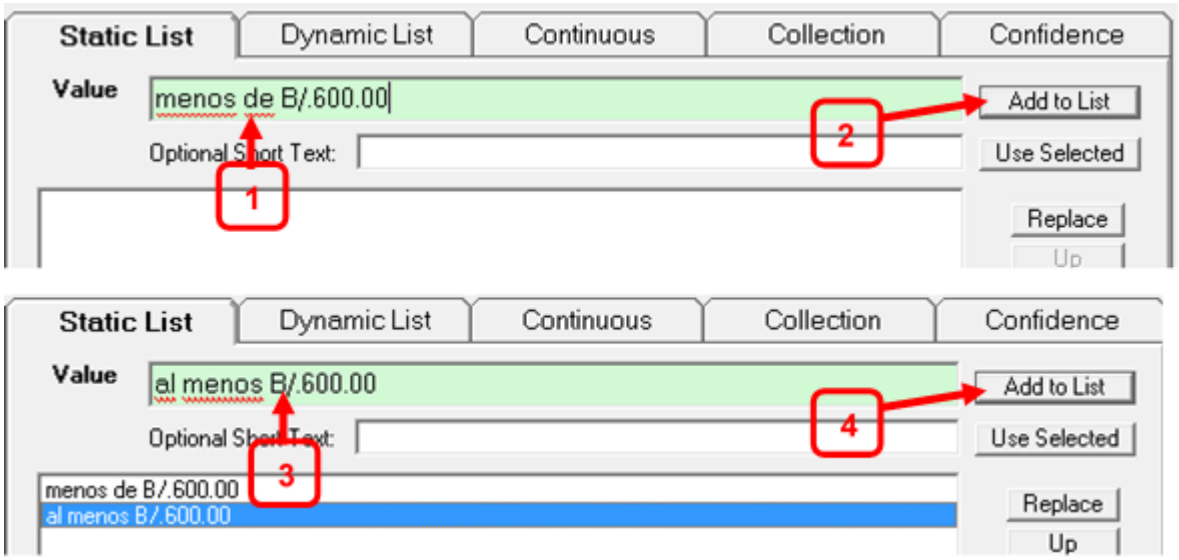


Figura 13. Ingreso de los valores para la variable Ingreso_Cliente

En este sistema se requiere que esta variable solo pueda tener un valor a la vez, por tanto, en el cuadrante donde se encuentra la pestaña **Prompt,** se activa la pestaña **Options** en la que se debe marcar la opción **SINGLE value** (Figura 14)**.**

Figura 14. Configuración adicional variable Ingreso_Cliente

Las siguientes variables son del tipo **Confidence** y sus nombres serán **"Préstamo_Aprobado"** y **"Préstamo_Denegado".** Al igual que con la primera variable que se creó, debe seleccionarse el botón **New** en la ventana de **Variables.** Luego se ingresa el nombre de **"Préstamo_Aprobado"** en el campo **Name**, pero esta vez se marca la opción **Confidence** en la ventana de **New Variable.** Seguido a esto se presiona **OK.** El **Prompt** para esta variable es **"El préstamo ha sido aprobado"** (Figura 15)**.**

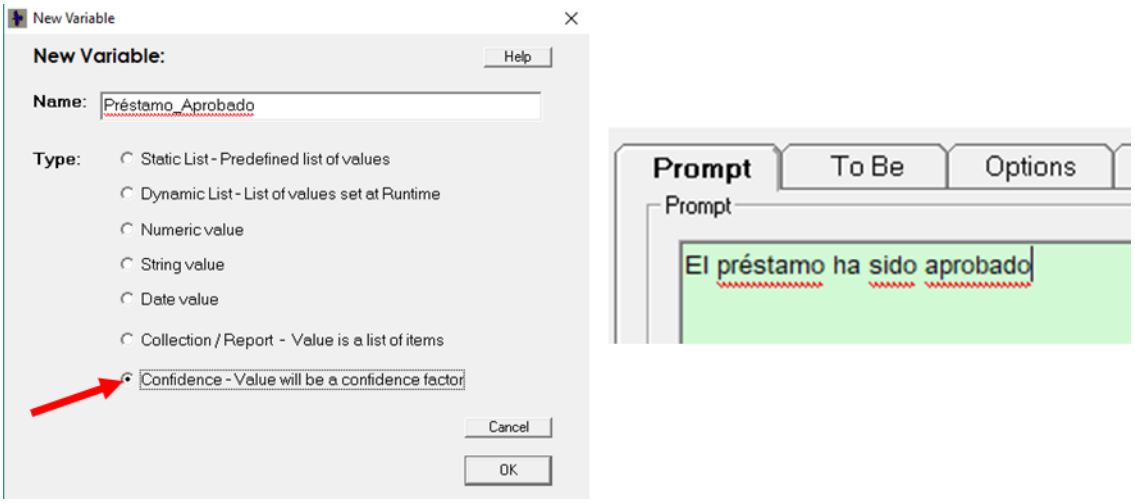


Figura 15. Creación variable Préstamo_Aprobado

Se realiza un proceso similar para la variable **"Préstamo_Denegado"** con su **Prompt** correspondiente **"El préstamo ha sido denegado"** (Figura 16)**.**

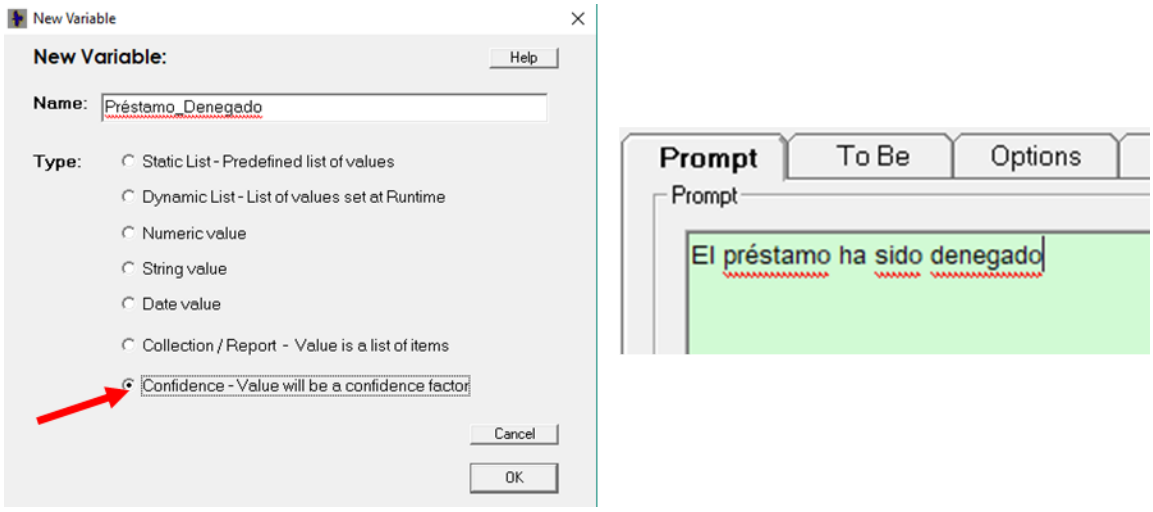


Figura 16. Creación variable Préstamo_Denegado

Con esto finaliza el proceso de creación de las variables con sus respectivos valores. Se hace clic en **OK** en la ventana de **Variables.**

2.2.4.3. Paso 3: Añadir los bloques lógicos (reglas)

Para añadir las reglas que serán evaluadas por el sistema se debe hacer clic en el icono de la **L** en la barra de herramientas de Corvid (Figura 17), lo que hará que aparezcan dos ventanas: **Logic Block** y **Rule View.**

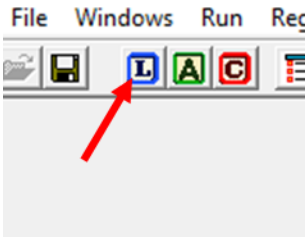


Figura 17. Icono para agregar bloques lógicos.

Para agregar una nueva regla es necesario hacer clic en la opción **Add** dentro del cuadrante **IF-AND** de la ventana **Logic Block.** Esto hará que se abra una nueva ventana **Add to Block.**

Dentro de esta ventana, se selecciona la variable **"Ingreso_Cliente"** y luego en la pestaña **Static List,** clic en la opción **Add Each Individually** y por último el botón **Done** (Figura 18).

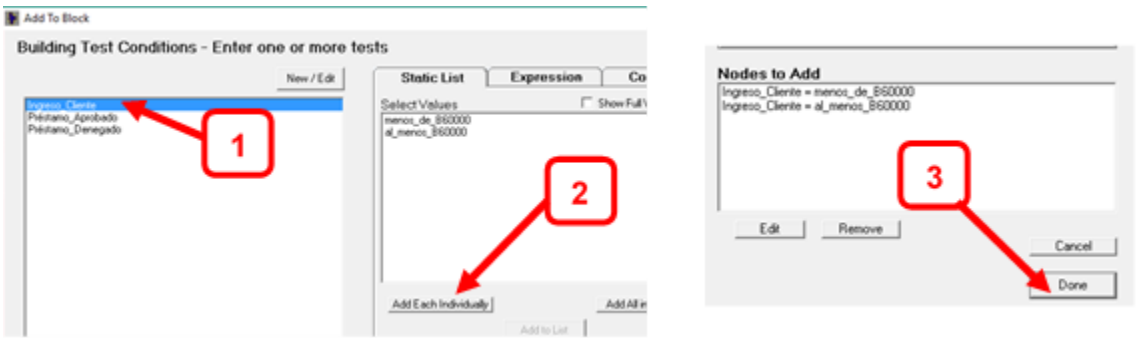


Figura 18. Ingreso de la primera regla del sistema

De esta forma, las reglas aparecen en la ventana **Logic Block.** Luego se selecciona la primera regla y se procede a hacer clic en el botón **Variable** del recuadro **THEN** (Figura 19)**.**

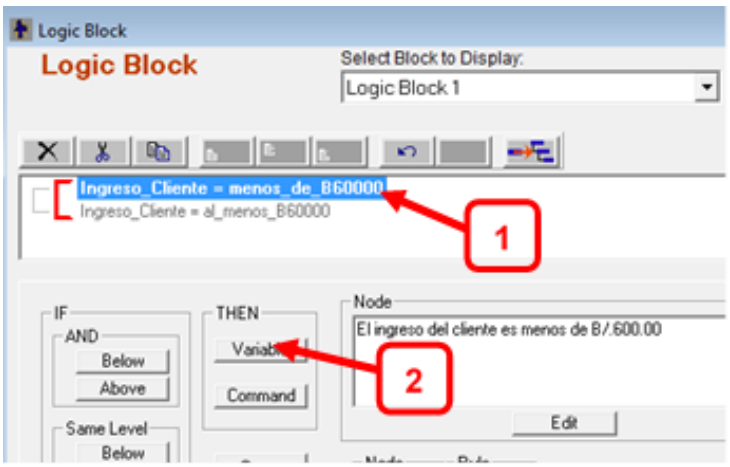


Figura 19. Adición de acción THEN.

Luego se escoge la variable **"Préstamo_Denegado"** y se le asigna un valor de confidence de **10.** Se hace clic en **Add to List** y luego en **Done** (Figura 20).

Figura 20. Edición regla para préstamo denegado.

Se repite el proceso para la siguiente regla y la variable **"Préstamo_Aprobado"** (Figura 21)**.**

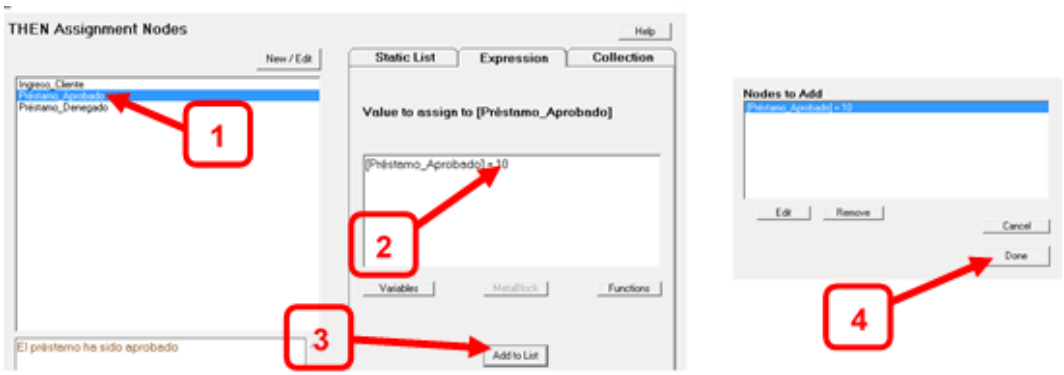


Figura 21. Edición regla para préstamo aprobado

Para finalizar esta etapa se da clic en el botón **Done** de la ventana **Logic Block** (Figura 22)**.**

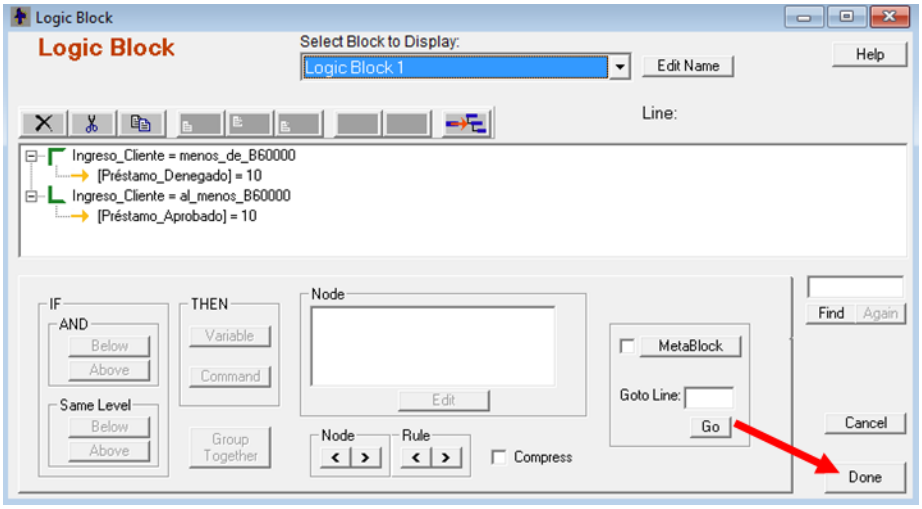


Figura 22. Finalización del proceso de añadir reglas.

2.2.4.4. Paso 4: Definir los bloques de comando básicos

Los sistemas creados en Corvid requieren como mínimo un bloque de comando para lograr que se ejecute. Para llevar a cabo esto, se debe dar clic en el icono de una **C** en la barra de herramientas (Figura 23), con lo cual se desplegará la ventana de **Command Block.**

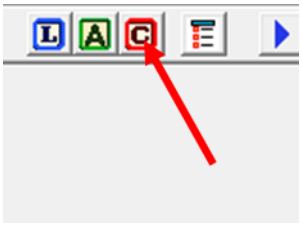


Figura 23. Icono de los bloques de comando.

Dentro de esta nueva ventana se da clic en el botón **Add**, lo que hará que se muestra otra ventana llamada **Commands**, en la cual se marcará la opción **All Confidence Variables** y luego se pulsa el botón de **OK** (Figura 24)**.**

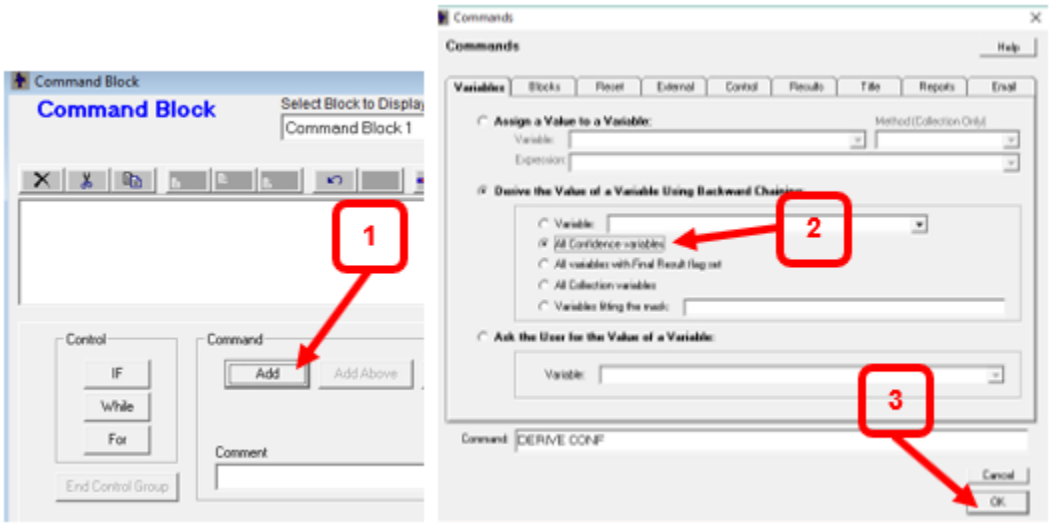


Figura 24. Inserción de bloque de comando Derive Conf

Ahora se pasa a agregar el comando **Results.** En la ventana **Command Blocks** se ha habilitado la opción **Add Below**, se da clic sobre ella y nuevamente se abre la ventana de **Commands,** en donde esta vez se usará la pestaña **Results** y se marcará la opción **Display Default Results Screen** (Figura 25)**.** Finalmente se da clic en **OK.**

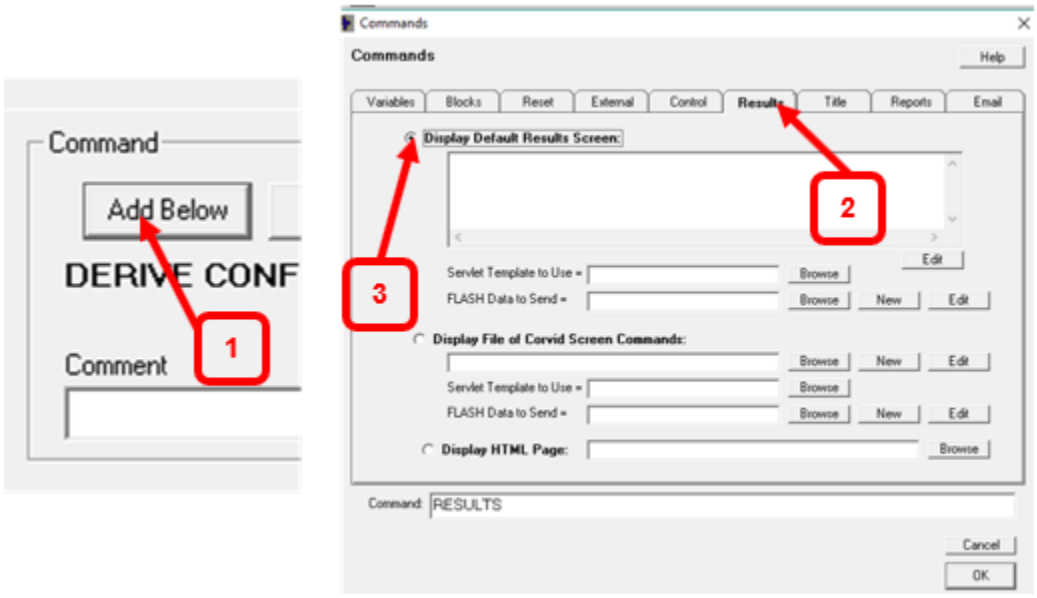


Figura 25. Inserción de comando Results

2.2.4.5. Paso 5: Ejecutar el sistema

El paso final consiste en ejecutar y probar el sistema creado, para lo cual, solo basta con dar clic sobre el triángulo azul que aparece en la barra de herramientas que se muestra en la figura 26.

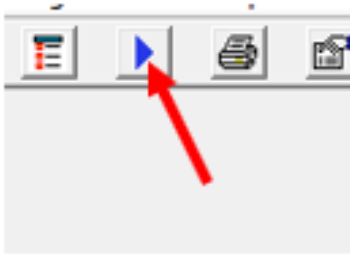


Figura 26. Icono para la ejecutar el sistema.

El sistema debe ejecutarse utilizando el navegador del equipo para lo cual es necesario haber realizado previamente las configuraciones pertinentes para el funcionamiento del sistema. En la figura 27 se muestra una captura del sistema, creado en este ejemplo, en su fase de ejecución.

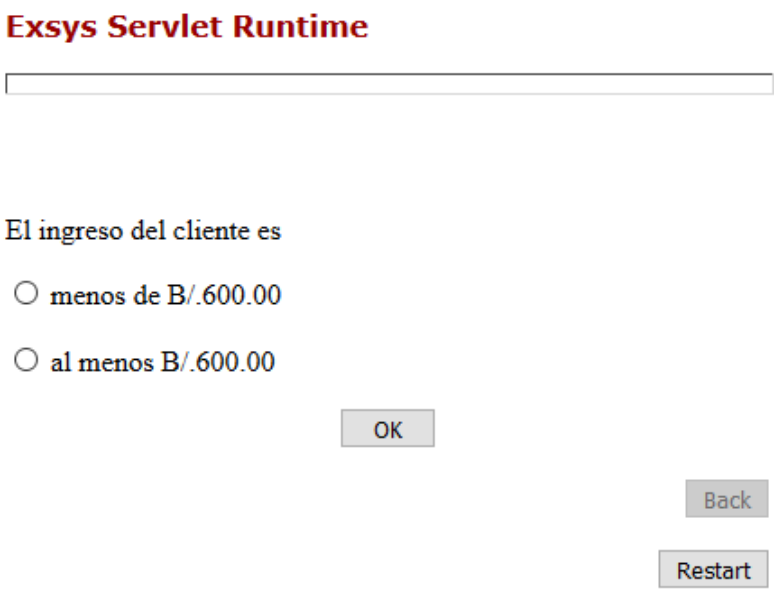


Figura 27. Sistema en ejecución

2.3. Prolog

Prolog es un lenguaje de programación del tipo descriptivo que está enfocado en la declaración de objetos y las relaciones que hacen ciertas una solución para un problema, lo que quiere decir que su objetivo es la descripción del conocimiento que se tiene sobre el problema y no la secuencia de pasos necesarios para resolverlo (Clocksin & Mellish, 2012). De esta forma, para un programa escrito en Prolog, la computadora se encarga de inferir el conocimiento a partir del que ha sido declarado previamente por el programador.

La base de la sintaxis de un programa en Prolog (Badaró et al., 2013), incluye:

a) Declaración de hechos sobre los objetos y sus relaciones.
b) Realización de preguntas sobre los objetos y sus relaciones.
c) Definición de reglas sobre los objetos y sus relaciones.

El lenguaje Prolog ha sido implementado en una gama de herramientas dedicadas a la creación de sistemas expertos, no obstante, para efectos del presente escrito se hará un enfoque en la herramienta Visual Prolog para la definición de las secciones que se encuentran en un programa de esta índole, no obstante existen herramientas como SWI Prolog, que además de facilitar un entorno para el

desarrollo de programas con este lenguaje, ofrece opciones de compilación adicionales, como la generación de ejecutables independientes y la posibilidad de ser combinado con el lenguaje C (González, 2007).

2.3.1. Directivas de compilación

Las directivas de compilación se refieren a las características del compilador. Son opciones que pueden modificarse desde el menú, líneas de comando o dentro del propio código fuente del programa que se esté trabajando.

Entre las características modificables del compilador se encuentran:

1. Tipo de controlador gráfico.
2. Tipo de fuente para un entorno MS-DOS basado en gráficos BGI.
3. Identificación de cláusulas no deterministas.
4. Especificación del tamaño del array interno.
5. Despliegue de mensajes de error durante la compilación.
6. Evitar la generación de código examinador.
7. Alertas de errores de sintaxis.
8. Habilitación de programación modular.

Una recomendación al momento de desarrollar un programa empleando el lenguaje Prolog, es salvar los cambios que se vayan realizando en el programa a medida que se avance en el proceso; además de la necesaria compilación antes de correr un programa que se acaba de modificar.

Los programas escritos en Prolog siguen una estructura definida mediante secciones que se verán a continuación.

2.3.2. Sección de constantes

El papel de las constantes dentro del lenguaje Prolog es similar al que ejercen en otros lenguajes de programación, debido a que se emplean para representar cantidades, símbolos, entre otros.

Esta sección se encabeza con la palabra reservada **Constants** y las variables de este tipo se declaran **<Nombre> = <Valor Asignado>**.

Para mostrar un ejemplo de la declaración de esta sección se tiene el caso de una cantidad de 10 autos, lo cual siguiendo la sintaxis de Prolog, queda de la forma:

```
CONSTANTS

Autos = 10
```

Sin embargo, el uso de constantes presenta un conjunto de limitaciones:

1. No existe distinción entre mayúsculas y minúsculas.
2. No es posible realizar definiciones recursivas entre constantes.
3. Toda constante debe ser definida antes de ser utilizada.
4. Sólo se pueden declarar una vez.

2.3.3. Sección de dominios

La palabra dominio dentro del entorno de Prolog se refiere a los tipos de datos de las variables que serán utilizadas en el programa, específicamente en los predicados. Prolog proporciona tipos de dato por defecto, no obstante, el programador puede personalizar dominios para hacer su programa más comprensible, es decir, puede crear nuevos tipos de variables basados en los existentes en Prolog de manera que los predicados se tornen más específicos en cuanto a la información que están manejando. Esta opción es útil cuando los programas son complejos y que es altamente probable que requieran mantenimiento luego de un plazo de tiempo.

Para tener más claro el concepto, un tipo de dato por defecto de Prolog es **symbol** Este tipo de dato es comúnmente para almacenar cadenas de texto. Si se requiere representar “Juan es asistente” se tendría un predicado así:

```
Persona(symbol, symbol)
```

Este predicado no da mucha idea de qué información sobre una persona se está representando, por tanto, mediante el uso de la sección de dominios se mejora este aspecto:

```
DOMAINS

Nombre, cargo = symbol
```

```
PREDICATES

Persona(Nombre, cargo)
```

La palabra reservada para esta sección es **Domains**. En el ejemplo se aprecia a simple vista que el predicado se vuelve más claro y específico.

Un aspecto importante que resaltar es que a pesar de que ambos, tanto nombre como cargo, son del dominio symbol, estos no son equivalentes, es decir, no comparten el mismo subdominio.

2.3.4. Sección de la base de datos

Como se ha mencionado en puntos anteriores, los programas pueden variar en su grado de complejidad, incluso llegando al punto en el que es imprescindible realizar modificaciones o actualizaciones en las bases de hechos creadas inicialmente, durante el tiempo de ejecución del sistema; para esto se encuentra a disposición la sección que emplea la palabra reservada **Facts** o **Database**.

La sintaxis para una sección de base de datos es la que se muestra a continuación:

```
[Global]{FACTS|DATABASE}-[nombre_bd]

[nocopy][{nondeterm|determ|single}] hecho_1[Argumentos]

…

[nocopy][{nondeterm|determ|single}] hecho_N[Argumentos]
```

En la sintaxis se observan otros elementos adicionales al nombre la base de datos y los hechos en sí, que tienen las siguientes funciones:

1. Indicar si la sección será global o no para el proyecto (Global).
2. Indicar si los datos generados no serán almacenados en la pila global del programa (nocopy).
3. Permitir la existencia de varias instancias para un hecho (nodeterm).
4. Especificar la existencia de una instancia para un hecho en el momento (determ).

5. Prohibir la existencia de más de una instancia para un hecho (single).

2.3.5. Sección de los predicados

La sección de los predicados debe ir precedida por la palabra **Predicates**. Está dedicada a establecer el tipo de argumentos que utilizarán las cláusulas, por ende, es imprescindible que tanto los predicados como las cláusulas se encuentren estrechamente relacionadas, de lo contrario el sistema no podrá discernir ni tratar la información que se le está suministrando. En pocas palabras, en esta sección se definen las relaciones entre los objetos.

Los tipos de argumentos utilizados en los predicados deben haber sido previamente definidos en la sección Domain.

En base a lo descrito se afirma que la sección de predicados está destinada a listar los predicados y los tipos de argumentos (dominios) que estos utilizan.

Un predicado consta de dos partes: el nombre y los argumentos.

Algunas restricciones para la selección de los nombres de un predicado son las siguientes:

1. Está prohibido el uso de caracteres no alfanuméricos como asteriscos, guiones, entre otros.
2. No se deben dejar espacios en blanco, en su lugar usar guión bajo (_).
3. El nombre debe iniciar con una letra minúscula.
4. La longitud máxima no debe exceder los 250 caracteres.

Algunos ejemplos de predicados:

```
DOMAIN

color_pared, color_cerca, nombre = string

edad = integer

PREDICATES

casa(color_pared, color_cerca)
```

```
persona(nombre, edad)
```

Otro ejemplo básico se mostró en la sección 2.3.3. de este documento.

2.3.6. Sección de cláusulas

Esta sección es el centro del programa en Prolog, pues es donde se encuentran las bases de datos y las reglas que serán analizadas para satisfacer las metas establecidas. Se encabeza con la palabra **Clauses.**

Una regla tiene la siguiente sintaxis:

```
Head:- <sub goal>, <sub goal>.
```

Ejemplo 1:

```
hijode(A,B) :- padrede(B,A).
```

Una regla se define cuando es necesario que se cumplan determinadas condiciones para la obtención de un resultado. En una regla la parte izquierda de la sentencia es cierta, sólo si se cumple la parte de derecha. Ambas partes de la regla se dividen mediante un "**:-**" que hace una analogía con la estructura "IF/THEN" de otros lenguajes de programación;además, deben finalizar con un punto (.).

En términos del ejemplo 1, esto significa que A es hijo de B si B es padre de A. Otro ejemplo de regla es el siguiente:

Ejemplo 2:

```
abuelode(A,B) :- padrede(A,C),padrede(C,B).
```

Siguiendo lo expuesto en el ejemplo 1, esta regla se lee de la forma: A es abuelo de B si A es padre de C y C es padre de B.

2.3.7. Sección de meta u objetivo

Finalmente se tiene la sección de metas que Prolog debe satisfacer. Una meta u objetivo comparte una estructura similar a la de una regla, excepto por la ausencia del símbolo "**:-**". Una meta puede estar compuesta de varias submetas.

2.3.8. Ejemplo: Uso de las secciones básicas de un programa en Prolog

Para efectos didácticos se procede a desarrollar un ejemplo básico en el que se haga uso de las principales secciones de un programa escrito en Prolog.

Se tiene la siguiente información sobre aves corredoras:

- Son aves corredoras el avestruz, emu y ñandú.
- El avestruz tiene patas largas. En cada pata tiene 2 dedos.
- El emú tiene patas fuertes. En cada pata tiene 3 dedos.
- El ñandú tiene patas largas. En cada pata tiene 3 dedos.

Se desea que el sistema identifique las aves que cumplen con la siguiente descripción:

1. Patas largas y dos dedos.

De forma adicional se solicita que se suministren las características conocidas del ñandú.

Una forma de solucionar este ejemplo es la siguiente:

```
%DECLARACIONES

ave_corredora(avestruz).

ave_corredora(emu).

ave_corredora(ñandues).

tipo_pata(patas_largas, avestruz).

tipo_pata(patas_fuertes, emu).

tipo_pata(patas_largas, ñandues).

cantidad_dedos(dos_dedos, avestruz).

cantidad_dedos(tres_dedos, emu).

cantidad_dedos(tres_dedos, ñandues).

%REGLAS
```

```
/* Se hace uso de la variable X que tomará el valor del nombre del ave que cumpla con los criterios definidos. La regla se lee de la siguiente manera: X tiene dos dedos y patas largas si X tiene patas largas y X tiene dos dedos*/

dos_largas(X):-

tipo_pata(patas_largas, X), cantidad_dedos(dos_dedos, X).

/* Para las características del ñandú, se usan dos variables X y Y. En donde las condiciones están diseñadas para transmitir la información almacenada en las declaraciones */
/* En la relación tipo_pata, X debe recibir el valor almacenado para el objeto ñandúes. De forma similar Y debe recibir el valor de la relación cantidad_dedos para el objeto ñandúes */

car_ñandues(X,Y):-

tipo_pata(X, ñandues), cantidad_dedos(Y,ñandues).
```

2.4. AgentBuilder

AgentBuilder consiste en un conjunto de herramientas integradas en un software que permite a los desarrolladores un avance más rápido y fácil en los proyectos de creación de software inteligentes debido a que reduce el tiempo y el costo a la vez que incrementa el rendimiento de sistemas inteligentes robustos (Acronymics Inc., 2004).

Entre sus características más resaltables se encuentran:

a) Fácil creación de agentes inteligentes.
b) No requiere conocimientos especiales en diseño de sistemas inteligentes ni conexión a la red.
c) Utiliza un lenguaje de alto nivel orientado a la programación de agentes.
d) Permite la construcción de agentes con capacidades de operación autónomas y para la comunicación con otros agentes y usuarios.

e) Entorno de desarrollo gráfico.
f) Está basado en Java, lo que lo convierte en un generador de aplicaciones multiplataforma.
g) Facilidad de integración con librerías en Java, C y C++.

Este software provee una gama de herramientas gráficas diseñadas acorde a las fases de construcción de un sistema inteligente. Estas incluyen:

a) Organización y control del desarrollo del proyecto.
b) Análisis del dominio del problema
c) Definición de la arquitectura del sistema.
d) Especificación del comportamiento del sistema.
e) Generación de ejecutables.
f) Revisión y depuración del sistema.

En la figura 28 y 29 se observa la apariencia y cuadros de diálogo de las herramientas que vienen integradas en el AgentBuilder para el proceso de desarrollo de sistemas inteligentes.

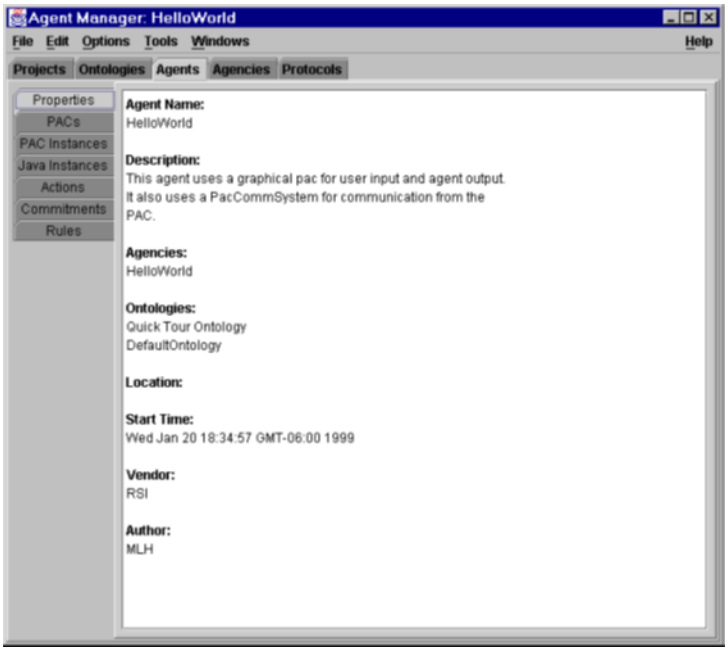


Figura 28. Ventana Project Manager de AgentBuilder - Tomado de (Acronymics Inc., 2004).

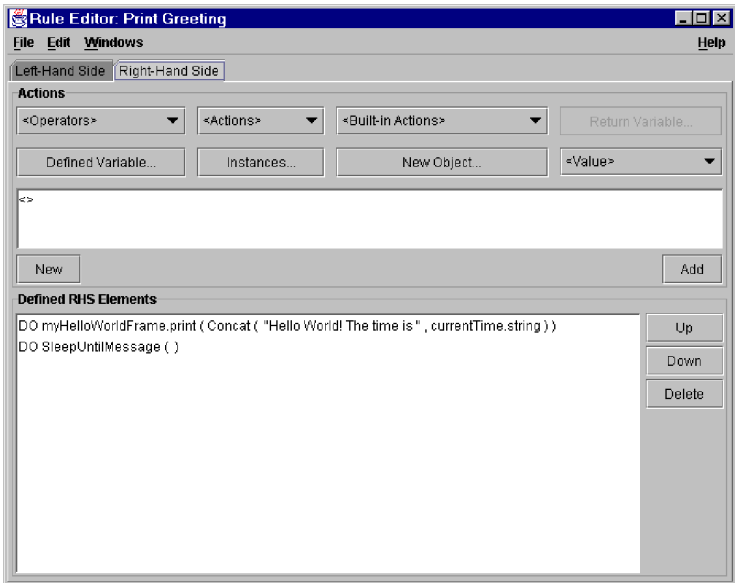


Figura 29. Ventana Rule Editor de AgentBuilder - Tomado de (Acronymics Inc., 2004).

Como se puede apreciar, el entorno se encuentra en el idioma Inglés, no obstante, su presentación hace que su uso sea intuitivo.

Actualmente AgentBuilder es producido en dos presentaciones, Pro y Lite; sin embargo, ambas son de pago.

2.5. Clips

C Language Integrated Production System (CLIPS) es un entorno de programación para el desarrollo de sistemas expertos. Fue creado en 1986 por Software Technology Branch (STB), Nada/Lyndon B. Johnson Space (Giarratano, 2017). En sus primeras versiones solo se encontraba en la capacidad de manejar hechos y reglas. Posteriormente a partir de la versión 6.0 existe la posibilidad de incluir objetos en las cláusulas de las reglas.

Es un entorno dirigido por datos, en otras palabras, los hechos o las instancias de los objetos son los responsables de estimular la ejecución del programa mediante su motor de inferencias.

Hay tres maneras para representar el conocimiento dentro de CLIPS:

1. Reglas: primordiales para la heurística aplicada al conocimiento.
2. Funciones: necesarias para el procesamiento del conocimiento.

3. Programación orientada a objetos: aplicación de clases, abstracciones, encapsulamiento, herencia y polimorfismo.

Componentes principales de un programa básico en CLIPS:

1. **Hechos**: representan información sobre el estado del mundo.

Para crear un hecho dentro de CLIPS se emplea el comando **assert**, de la siguiente manera:

```
CLIPS> (assert fruta manzana)
```

Este comando obtendrá como respuesta:

```
<fact-0>
```

La respuesta fact-0 corresponde al identificador del hecho. El dígito que aparece en el identificador es único, esto quiere decir que, a pesar de que un hecho se elimine al añadir un nuevo hecho, este tomará un nuevo identificador y no el que fue eliminado.

El comando para eliminar un hecho es **retract f-id**, es significa que, para eliminar un hecho, es necesario conocer el identificador de este. El comando para listar todos los hechos es **facts**.

Al momento de crear hechos, hay que poner especial atención en el uso de mayúsculas y minúsculas, pues el entorno es sensible a estas; por lo que debido al uso de estas se da origen a hechos diferentes.

La creación de hechos por sí solos tienen evidentes limitaciones, por lo que se requiere construir funciones que les den utilidad.

2. **Reglas**: establecen las acciones a realizar con / sobre los hechos.

Por su parte las reglas en CLIPS constan de tres partes:

a) defrule: comando con el que se asigna un nombre único a la regla.
b) Condición: equivalente al segmento IF.
c) Acción: equivalente al segmento THEN.

En base a lo dicho, así es la estructura de una regla en CLIPS:

```
(defrule pato (animal-es pato) => (assert (sonido-es cuack)))
```

3. **Agenda**: conjunto de reglas.

Para trabajar un programa en CLIPS hay a disposición dos modalidades, mediante ventana de comandos o a través de una GUI (Figura 30).

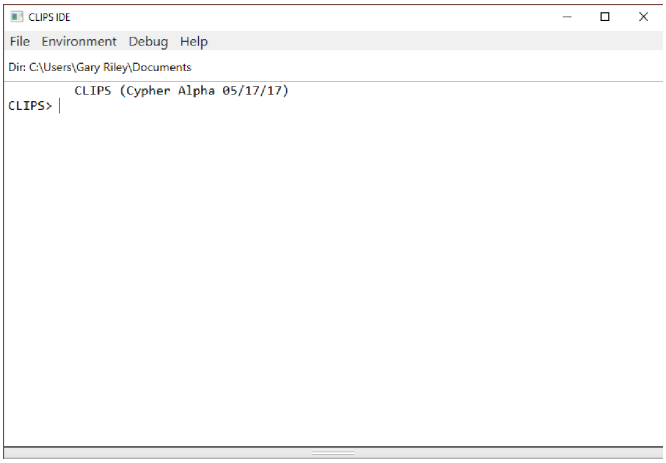


Figura 30. IDE Clips para Windows tomado de (Riley, 2017)

2.6. Jess

Java Expert System Shell (JESS) es una variante de CLIPS escrito puramente en Java. JESS tiene dos usos fundamentales de los que destaca su papel como máquina de reglas. Un programa basado en reglas llega a tener hasta cien o incluso miles de reglas que JESS puede aplicar continuamente a los datos almacenados en forma de base de conocimiento. El algoritmo que usa es una versión mejorada de Rete para emparejar las reglas con la base de conocimiento (Friedman-Hill, 2003).

Dos características fundamentales de JESS:

1. Usa el método de encadenamiento hacia adelante para la inferencia del conocimiento.
2. El modo de ingreso de comandos sólo es para el desarrollador y no para el usuario final.

Es posible trabajar con JESS en aplicaciones de líneas de comando, aplicaciones con interfaz gráfica de usuario, servlets y applets.

Componentes básicos de un programa en JESS (Friedman-Hill, 2003):

a) **Identificadores y tipos de datos:**
    a. Atoms: Consisten en símbolos, nombres creados por el usuario para determinados elementos dentro del programa. Incluyen elementos como caracteres no alfanuméricos ($, *, =, etc.) y no pueden iniciar con un número o signo de puntuación; además de ser sensibles a las mayúsculas y minúsculas.
    b. Numbers: Incluyen los tipos enteros y puntos flotantes.
    c. Strings: Se denotan entre comillas ("…") y los saltos de líneas deben ser indicados en forma real, es decir, no trabaja como el Java puro en donde un salto de línea se denota con "\n".
    d. Lists: Son un conjunto de elementos de un mismo tipo o diferentes.
    e. Comments: Los comentarios se denotan con punto y coma (;).
b) **Funciones**: Jess tiene integradas una gama de funciones para realizar operaciones matemáticas, controlar el programa y manejo de cadenas.
c) **Variables**: Las variables dentro de JESS corresponden a atoms, cuyos nombres inician con un signo de interrogación (?), por lo tanto, este signo forma parte del nombre. Cada variable puede o no ser declarada antes de su primer uso, a excepción de las variables globales.
d) **Deffunctions**: Tal como otros entornos, JESS permite la creación de funciones por parte del usuario, para lo que se usa el comando **deffunction** seguido del nombre de la función (atom) y los argumentos (variables).
e) **Defadvice**: Comando empleado para añadir o extender el código de las funciones propias de JESS, lo que da la opción de un mayor control sobre el comportamiento del programa.
f) **Java reflection**: Opciones que brindan la posibilidad de emplear o manipular objetos directamente desde Java dentro de JESS.

g) **Base de conocimiento**: Conformado por un conjunto de hechos. Existen tres categorías de hechos en JESS: ordered, unordered and deffacts.

h) **Defrules**: Al ser una variante de CLIPS, el método para la creación o definición de reglas es similar (Punto 2 de los componentes básicos de un programa en CLIPS).

i) **Defqueries**: Poseen una estructura similar a las reglas, a excepción del lado derecho. Se utilizan para buscar el conocimiento dentro de la base del programa.

j) **Defmodules**: Recurso empleado para facilitar el desarrollo de programas altamente complejos debido a que ofrece la alternativa de subdividir el programa en segmentos más pequeños.

3. INTRODUCCIÓN A LA METODOLOGÍA COMMONKADS

El término metodología hace referencia un conjunto de pasos o fases que se siguen con el fin de alcanzar un propósito; además que ofrece una estructura y mayor control, lo que facilita el desarrollo de un proyecto y su gestión.

Un SBC es un software y como tal, es necesario aplicar una metodología para llevar a cabo su correcto desarrollo, proceso que no debe desligarse de la generación de documentación para efectos de comprensión de su funcionamiento con el objetivo de implementar modificaciones que incrementen su rendimiento y eficacia, en el momento que se amerite; sin embargo, la Ingeniería de Software por sí misma no contempla rasgos necesarios para la construcción de SSBBCC, debido a esto, se da lugar al establecimiento de una nueva disciplina: la Ingeniería del Conocimiento (Studer, Benjamins, & Fensel, 1998).

La ingeniería del conocimiento, además de enfocarse en la construcción de SSBBCC, también vela por el análisis y procesos de mantenimiento de estos a su vez que desarrolla métodos, lenguajes y herramientas especializadas para el diseño de este tipo de sistemas (Studer et al., 1998).

3.1. Aspectos generales de la Metodología CommonKADS.

Originalmente llamada Knowledge Acquisition Design System (KADS) para luego de su éxito, ser ampliando y adoptar el nombre CommonKADS, es una metodología enfocada en la construcción de SSBBCC, como lo son MIKE y Protégé. Es el resultado de varios proyectos dentro del programa ESPRIT, que buscaba la innovación y aplicación de la informática avanzada en la Unión Europea, de manera que se obtuviera una calidad industrial en los SSBBCC de forma estructurada, manejable y repetitiva (Schreiber et al., 2001). Esta metodología se desarrolló en la Universidad de Ámsterdam junto a socios europeos (otras universidades, organizaciones de investigación y consultoría).

Debido a la cantidad de SSBBCC que han sido desarrollados empleando esta metodología, es considerada como un estándar por diversas compañías y organizaciones en el mundo.

3.1.1. Principios fundamentales de la Metodología CommonKADS.

La metodología CommonKADS provee de una aproximación estructurada para el diseño de un SBC. Esta metodología sigue unos principios que a medida que pasaron los años fueron ganando mayor fortaleza. Estos principios se detallan a continuación:

1. *La ingeniería del conocimiento consiste en construir diferentes modelos para la representación de ciertos aspectos del conocimiento humano.*

La visión tradicional de la Ingeniería del conocimiento como un proceso de extracción del conocimiento para ser implementado en un sistema computacional ha quedado como una perspectiva superficial de esta disciplina. La ingeniería del conocimiento se reconoce como una actividad de modelado, de construcción de una descripción de los aspectos fundamentales del conocimiento, ignorando otros elementos que no poseen relevancia dentro del mismo.

2. *El principio del nivel de conocimiento: lo primero es concentrarse en los aspectos conceptuales del conocimiento para entonces proceder con la programación.*

El aspecto más importante en el desarrollo de un SBC se encuentra del lado humano, por lo que se requiere estudiar a profundidad la situación real del conocimiento considerando tanto expertos, usuarios y el comportamiento que debe tener dentro de la organización en la que será implementado.

3. *El conocimiento posee una estructura interna estable en la que se distinguen tipos de conocimiento específicos y sus roles.*

Este principio hace referencia a que el conocimiento puede ser complejo mas no caótico, pues es altamente probable que se encuentren categorías genéricas, patrones y ciertas estructuras que permitan comprender su papel en la solución de problemas.

4. *Un proyecto de conocimiento debe ser gestionado basado en el aprendizaje de las experiencias propias en un controlado estilo de espiral.*

CommonKADS es una metodología más flexible que el tipo en cascada y más controlable que el prototipado rápido. Esto es así ya que el conocimiento es demasiado rico y difícil de encajar en metodologías con aproximaciones estrictas.

3.1.2. Ciclo de Vida de la metodología CommonKADS.

El principio 4, tratado en la sección anterior, hace mención de que los proyectos en lo que se manipula conocimiento, deben seguir un estilo de espiral para su desarrollo, debido a la complejidad del conocimiento. No obstante, se siguen protocolos básicos de cualquier metodología:

a) División del proceso en fases.
b) Cada fase consta de actividades diferentes.
c) Al final de cada fase se debe generar un producto.

Se reconocen las siguientes etapas en el proceso de producción de un SBC (Henao Cálad, 2001):

1. Análisis: Su propósito es comprender el problema desde la perspectiva en la que se desea desarrollar. Productos característicos de esta fase: documento del proyecto en el que se incluyan: requerimientos, modelo conceptual, viabilidad y documentos de apoyo.
2. Diseño: Se elabora la descripción funcional (comportamiento) del sistema, así como una descripción física en la que se hace énfasis en sus componentes.
3. Implantación: Involucra la inserción de la aplicación dentro del contexto u organización para el que fue diseñado.
4. Instalación: El sistema inicia operaciones dentro del entorno en que fue implantado.
5. Uso: Se somete el sistema a ser manejado por los usuarios finales para la obtención de resultados.
6. Mantenimiento: Corresponde a la etapa de mejoras o actualización del conocimiento.

3.2. Conjunto de modelos de la metodología CommonKADS.

CommonKADS cuenta con una serie de formularios enfocados en rasgos específicos para el modelado del conocimiento. Estos formularios son agrupados en tres categorías de acuerdo a su contribución para responder las tres preguntas fundamentales en el proceso de modelado del conocimiento: "¿Por qué?", "¿Qué?" y "¿Cómo?" (Schreiber et al., 2001).

3.1.1. Nivel Contextual – Modelos de Contexto.

El nivel contextual se enfoca en responder la pregunta "¿Por qué?". De esta forma se debe aclarar el motivo por el cuál es necesario implementar un SBC para la solución de un problema. Se determinan los beneficios, costos y el impacto dentro de la organización o entorno. Para esto es imprescindible comprender el contexto organizacional y el entorno en el que será implementado el sistema.

En este nivel se encuentran los siguientes modelos:

a) Modelo organizacional
b) Modelo de tareas
c) Modelo de agentes

Estos modelos se verán con mayor detalle en el capítulo 4 de este documento.

3.1.2. Nivel Conceptual – Modelos Conceptuales.

La siguiente pregunta que debe responderse es "¿Qué?". El nivel conceptual va dirigido en determinar la estructura y naturaleza del conocimiento y la manera en que este es transmitido o intercambiado en el mundo real.

Los modelos encargados de esta misión son:

a) Modelo del conocimiento
b) Modelo de comunicación

Ambos modelos se tratarán con mayor profundidad en el capítulo 5.

3.1.3. Nivel Computacional – Modelo de Diseño.

La última pregunta que se debe responder es “¿Cómo?”, esto apoya al segundo principio que establece que es necesario definir primero un modelo conceptual para entonces proceder a la fase de programación del sistema. En este nivel se determina la manera en que se va a implementar el sistema, la arquitectura del software y los aspectos computacionales y técnicos.

El modelo destinado a esta labor es:

a) Modelo de diseño.

Se verá con mayores detalles en el capítulo 6 del presente documento.

En la figura 31 se observan los tres niveles de modelado del conocimiento y cómo están relacionados cada uno de sus componentes.

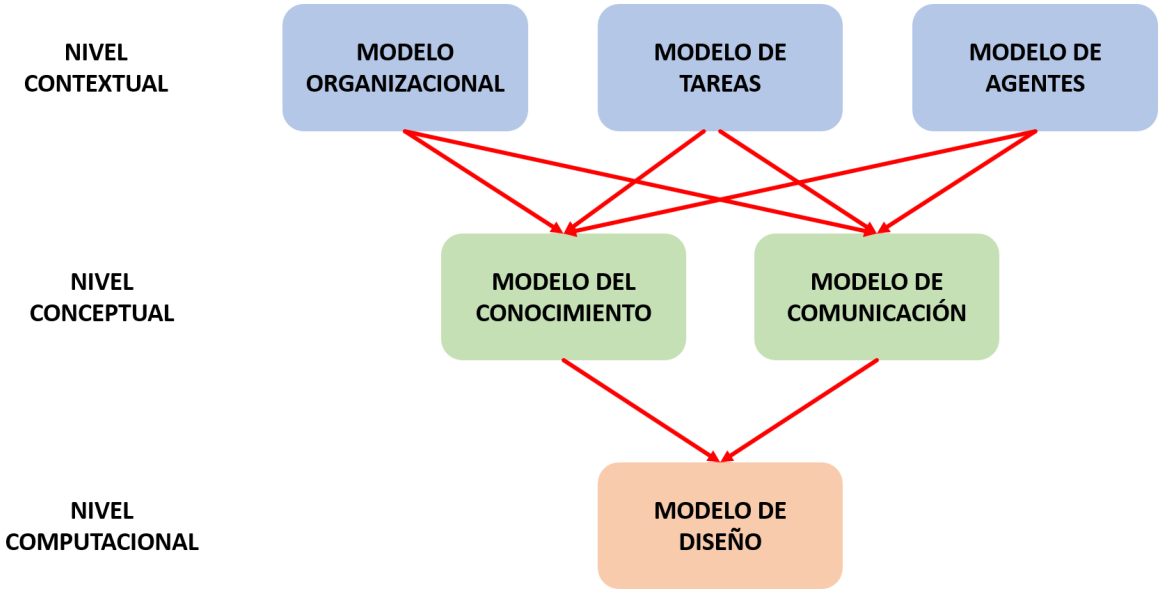


Figura 31. Niveles CommonKADS y sus componentes.

Se resalta el hecho que los formularios de la metodología CommonKADS contemplan elementos a nivel empresarial, no obstante, son adaptables de acuerdo con la complejidad del estudio que se esté realizando para la construcción de un SBC.

4. MODELADO DEL NIVEL CONTEXTUAL EN COMMONKADS

La primera etapa para el desarrollo de un SBC, siguiendo la metodología CommonKADS, abarca una serie de formularios enfocados en descubrir y formalizar aspectos relacionados al entorno en el que se planea insertar el SBC. Algunos de estos aspectos incluyen la identificación de problemas, oportunidades, soluciones y la factibilidad de estas.

El modelado a nivel contextual se encuentra subdividido en tres modelos: organizacional, tareas y agentes.

4.1. Modelo de la Organización de la metodología CommonKADS

Dentro del modelo organizacional se describe la estructura como tal del entorno especificando las funcionalidades de cada uno de sus elementos; proceso por el cual es probable detectar deficiencias como oportunidades para mejorar las distintas tareas con la implementación de un SBC (Studer et al., 1998). Esto es necesario debido a que un SBC es un agente que va a colaborar en el desarrollo de los procesos de la organización y para eso debe estar en las condiciones apropiadas para ser integrado en ese entorno (Schreiber et al., 2001).

Este modelo está conformado por un total de cinco formularios, identificados por las siglas OM (Organizational Model) y un índice que va del 1 al 5.

4.1.1. Descripción de los problemas y posibilidades de mejora.

El primer formulario, OM-1, persigue como fin principal el contexto de la organización, sus problemas y sus probables soluciones; estos últimos requieren una perspectiva lo más realista posible, así como un entendimiento concreto y explícito desde una vista de negocios.

Para la obtención de la información requerida por este formulario es propicio llevar a cabo entrevistas con los colaboradores de la organización, lluvia de ideas, reuniones con los gerentes, entre otras técnicas de recolección de información.

El personal entrevistado es clasificado en base a su interés dentro del sistema, obteniendo así a los denominados stakeholders del proyecto. Los stakeholders se

dividen en tres categorías para un proyecto de SBC: Proveedores del conocimiento, usuarios del conocimiento y personas encargadas de tomar decisiones basadas en el conocimiento.

Esta primera parte del modelado abarca elementos no variantes de la organización, como lo es la misión, objetivos y factores externos que influencian a la entidad.

En la tabla 2, se muestra la estructura básica del formulario OM-1 y una breve descripción de la información que se debe colocar en sus apartados.

Tabla 2. Formulario OM-1.

Modelo Organizacional	**Problemas y oportunidades OM-1**
Problemas y oportunidades	Lista de los problemas y oportunidades que se perciben, luego de aplicadas técnicas de recolección de información: entrevistas, lluvias de ideas, reuniones, discusiones con los gerentes, encuestas, etc.
Contexto Organizacional	Se indican las características claves del contexto organizacional: 1. Misión, visión y objetivos de la organización. 2. Factores externos que guardan relación con la organización. 3. Estrategias de organización. 4. Cadena de valores y controladores de valores.
Soluciones	Lista de las posibles soluciones a los problemas e ideas para aprovechar las oportunidades detectadas, de forma que queden enfocadas al contexto en el que se maneja la organización.

4.1.2. Descripción de los aspectos de interés de la organización.

El segundo estudio por realizar va de la mano con los elementos que poseen probabilidades de cambiar una vez se haya integrado el sistema en la organización, por ejemplo, el orden de los procesos, recursos, personal involucrado, entre otros. Es importante señalar, que el formulario OM-2 está diseñado para centrarse en una sola problemática-oportunidad de las que se

listaron en el OM-1. En caso de requerirse el análisis de más de un área, este formulario debe ser llenado de forma individual por área.

En la tabla 3 se exponen los elementos que deben considerarse en el estudio de elementos variantes de la organización frente a un SBC.

Tabla 3. Formulario OM-2

Modelo Organizacional	Aspectos Variantes OM-2
Estructura	Corresponde al organigrama de la porción de la organización que se ve involucrada con el problema / oportunidad en cuestión, en términos de departamentos, grupos, unidades, secciones, etc.
Proceso	Es el nombre que se le da de forma general a un conjunto de tareas. Representa un elemento relevante dentro de la estructura que se ha identificado.
Personal	Se indica al personal involucrado como proveedores, usuarios y personas que toman las decisiones en base al conocimiento. Estos pueden ser directores, consultores, etc.
Recursos	Detalle de los recursos utilizados en los procesos del negocio. Estos se clasifican en: 1. Recursos computacionales (sistemas de información). 2. Equipos y materiales. 3. Tecnología, patentes y derechos legales.
Conocimiento	Recurso especial, explotado por el negocio. En este apartado se identifica el conocimiento que es utilizado por la organización de forma breve.
Cultura y poder	Aspectos informales dentro de la organización: comunicación, habilidades, relaciones entre los miembros del personal y demás redes.

4.1.3. Descripción detallada de los procesos de interés.

En esta fase, se le da un vistazo más profundo al proceso que se esté estudiando. Un proceso es divisible en tareas más pequeñas y sencillas. Este proceso de análisis generalmente recurre al uso de diagramas de actividades basados en

Unified Model Language (UML), de igual forma este diagrama funciona como guía en la sección de procesos del formulario OM-2.

Un diagrama de actividades describe el estado de estas, mostrando la secuencia en que son realizadas. La mayor ventaja del uso de un diagrama de actividades para el análisis de los procesos reside en la posibilidad de ver de forma gráfica su flujo y de la información que en ellos transita. En la figura se muestra la notación básica de un diagrama de actividad.

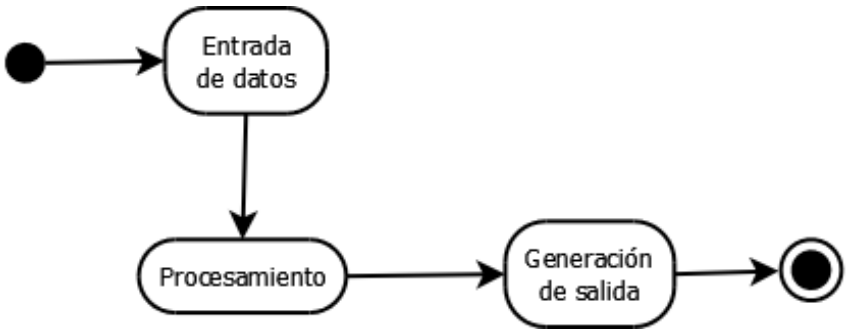


Figura 32. Ejemplo de la estructura básica de un diagrama de actividades.

El formulario correspondiente para mayores detalles de los procesos es el OM-3, permitiendo la posibilidad de estudiar a mayor profundidad las tareas que lo componen, en donde a menudo son necesarios ciertos cambios o combinar de forma diferente las tareas dentro de un proceso. Este formulario se muestra en la tabla 4. Se añaden filas de acuerdo con la cantidad de tareas que conforman el proceso bajo estudio.

Cada columna de este formulario se describe a continuación:

a. **No.:** Corresponde a un identificador, ya sea un número de tarea o algún código, que permita diferenciar cada tarea de forma individual.
b. **Tarea:** Es el nombre de la tarea.
c. **Encargado:** Agente encargado de ejecutar la tarea, puede ser un humano o un software.
d. **Lugar:** Ubicación de la tarea dentro de la estructura de la organización.
e. **Activo de conocimiento:** Listado de los recursos de conocimiento utilizados por la tarea.

f. **Intensivo:** Indicador booleano para identificar si la tarea hace uso intensivo de los recursos de conocimiento.
g. **Importancia:** Nivel de importancia de la tarea dentro de una escala numérica determinada.

Tabla 4. Formulario OM-3.

Modelo Organizacional		**Análisis del proceso OM-3**				
No.	Tarea	Encargado	Lugar	Activo de Conocimiento	Intensivo	Importancia
(a)	(b)	(c)	(d)	(e)	(f)	(g)

4.1.4. Descripción del componente conocimiento.

El componente de conocimiento es la pieza fundamental del estudio y es usado por las tareas al igual que por los trabajadores de la organización. Los aspectos fundamentales dentro de este apartado recaen en medir la dimensión del conocimiento, identificando las formas adecuadas para lograr su optimización en cuanto a presentación, accesibilidad y calidad. El formulario destinado a este estudio es el OM-4, tabla 5, y básicamente consta en preguntas simples cuyas respuestas suelen ser sí o no, y se añaden comentarios al respecto. Conserva algunos elementos del OM-3 como el nombre del activo del conocimiento, el agente que lo posee y para qué fin es usado.

Tabla 5. Formulario OM-4.

Modelo Organizacional		**Activos de conocimiento OM-4**				
Activo de conocimiento	Poseído por	Utilizado en	¿Forma correcta?	¿Lugar adecuado?	¿Tiempo correcto?	¿Calidad Adecuada?

4.1.5. Análisis de viabilidad.

El último paso dentro del modelo organizacional es el análisis de factibilidad, que no es más que un enfoque en los elementos principales de los cuatro formularios anteriores en los que se presta mayor atención en:

a. Áreas de aplicación más prometedoras y las mejores soluciones.
b. Beneficios versus costos.
c. Disponibilidad de la tecnología requerida.
d. Acciones que pueden impulsar el proyecto.

En la tabla 6 se muestra el formulario OM-5 en el que se detallan los puntos que se deben evaluar en el proyecto. Estos puntos se evalúan para cada área

Tabla 6. Formulario OM-5.

Modelo Organizacional	**Análisis de Factibilidad OM-5**
Factibilidad en el negocio	Se determinan: 1. Beneficios para la organización (tangibles económicamente, intangibles). 2. La magnitud de su valor. 3. Estimación de costos. 4. Comparación frente a otras alternativas. 5. Cambios organizacionales requeridos. 6. Riesgos económicos y comerciales ante la solución escogida.
Factibilidad técnica	Dentro de los aspectos técnicos para la realización del proyecto se evalúan los siguientes aspectos: 1. Complejidad, en términos de almacenamiento del conocimiento y el posterior proceso de razonamiento. 2. Existencia de elementos críticos como tiempo, calidad y recursos necesarios para el sistema. Cómo sobrellevar estos elementos críticos. 3. Medidas de éxito claras y métodos de validación, verificación de calidad y rendimiento para el sistema. 4. Nivel de complejidad de las interfaces de usuario.

	5. Nivel de complejidad en la interacción con otros sistemas de información. 6. Posibles riesgos e incertidumbres respecto al uso de determinada tecnología.
Factibilidad del proyecto	En cuanto al mismo proyecto es necesario saber: 1. Nivel de compromiso de los actores y stakeholders del sistema. 2. Disponibilidad de los recursos necesarios en términos de tiempo, presupuesto, equipo y personal. 3. Disponibilidad del conocimiento y otras competencias. 4. Realidad de las expectativas. 5. Adecuada comunicación interna y externa con la organización del proyecto. 6. Posibles riesgos e incertidumbres.
Acciones propuestas	Esta sección recolecta los resultados de las tres secciones anteriores, y es la que determina el proceso de la toma de decisiones con respecto a la realización del proyecto. Se recomiendan pasos de acción: 1. Determinar el enfoque. 2. En base al enfoque, determinar la solución apropiada. 3. Identificar los resultados, costos y beneficios esperados. 4. Acciones por ejecutar dentro del proyecto para alcanzar esos resultados. 5. Riesgos, definir planes de acción en caso de cambio de condiciones dentro o fuera de la organización.

El formulario OM-5 ha sido diseñado para el estudio de una sola problemática-oportunidad; por lo tanto, es necesario repetir el mismo análisis para cada área que se quiera estudiar dentro del proyecto.

Este es el último formulario empleado dentro de CommonKADS para el análisis organizacional.

4.2. Modelo de Tareas de la metodología CommonKADS.

El estudio siguiente al modelo organizacional, es el modelo de tareas, este tiene un enfoque en las características de las tareas relevantes dentro del sistema. Es una forma de refinar los resultados obtenidos en los formularios OM. Sin embargo, antes de proceder a realizar el análisis correspondiente, es importante contar con la certeza de que se han identificado correctamente las tareas dentro del proceso del negocio. Para esto, se tienen a continuación las características de una tarea:

a. Es una actividad orientada a una meta que añade valor a la organización.
b. Maneja entradas y genera las salidas deseadas de forma estructurada y controlada.
c. Utiliza recursos.
d. Requiere y provee conocimiento y otras competencias.
e. Ejecutada bajo estándares definidos y criterios de rendimiento.
f. La ejecuta un agente (humano o software).

Este análisis cuenta con dos formularios TM-1 y TM-2, el primero destinado a refinar los datos o resultados obtenidos en el OM-3; mientras que el segundo, va orientado en las posibles mejoras u optimizaciones que sean necesarias en la organización con respecto al manejo del conocimiento.

4.2.1. Descripción detallada de tareas.

Los formularios anteriores proveen una visión del conocimiento y las tareas orientadas al entorno del negocio, no obstante, en el modelo de tareas ya es más factible añadir información como dependencias en el flujo de la información, los objetos manipulados, el control del tiempo; a su vez que se aplica un análisis estructurado propio de la ingeniería de software y una metodología orientada a objetos. En la tabla 7 se muestra el formato del formulario TM-1 además de describir la información que debe ser colocada en cada una de sus secciones.

Tabla 7. Formulario TM-1.

Modelo de Tareas	Análisis de Tarea TM-1
Tarea	Se coloca el identificador (No.) y el nombre de la tarea.

Organización	Se indica a qué proceso pertenece la tarea y en qué parte de la organización es llevada a cabo.
Objetivo y valor	Descripción del objetivo de la tarea y el valor que su ejecución añade al proceso.
Dependencia y flujos	Tareas de entrada y tareas de salida. Es válido recurrir a un diagrama de actividades o diagrama de flujo para representar esta información.
Objetos manipulados	Los objetos de entrada y salida, se incluye paquetes de información y conocimiento. Los objetos internos, información o conocimiento, que use la tarea para su funcionamiento individual. Es funcional utilizar un diagrama de clases para describir la información que transita en la tarea.
Tiempo y control	Respecto al tiempo se define la frecuencia y duración de la tarea. Se añade una descripción de las condiciones (pre y post) para disparar o detener la ejecución de la tarea. El diagrama auxiliar en esta sección puede ser de estados o de actividades.
Agentes	Personal o software que ejecuta la tarea.
Conocimiento y capacidad	Capacidades, habilidades requeridas para la correcta ejecución de la tarea; así como las generadas por la tarea. Los objetos de conocimiento no se detallan en esta sección.
Recursos	Definición de los recursos utilizados por la tarea, preferiblemente en cantidades numéricas. Se incluyen: personal, sistemas, equipos, materiales y presupuestos.
Calidad y eficiencia	Listado de parámetros con los que la organización evalúa si la tarea se ha ejecutado correctamente.

4.2.2. Especificación del conocimiento.

La especificación del conocimiento es una ampliación del formulario OM-4, además de que se concentra en los cuellos de botella y las posibles mejoras que requiera el sistema con mayor detalle. Se realizan preguntas sencillas a los trabajadores y miembros que tengan contacto con el conocimiento. En la tabla 8 se muestra un ejemplo de la estructura del formulario TM-2.

Tabla 8. Formulario TM-2.

Modelo de Tareas	**Objeto de conocimiento TM-2**

Nombre	Nombre del objeto de conocimiento.
Poseído por	Agente
Usado en	Identificador y nombre de la tarea en la que se utiliza.
Dominio	Campo de especialidad, disciplina, rama de la ingeniería, etc.

Naturaleza del conocimiento		**Cuello de botella / Mejora**
Formal, Rigoroso		
Empírico, cuantitativo		
Etc.		
Forma del conocimiento		
Mente, papel, etc.		
Disponibilidad del conocimiento		
Tiempo, espacio, etc.		

4.3. Modelo de los Agentes de la metodología CommonKADS.

Dentro de los dos modelos anteriores se han estudiado las tareas desde la perspectiva de negocio y en la manera en que estas trabajan; por lo que en este último modelo se hace un paréntesis en los agentes encargados de realizar las tareas que ya fueron identificadas.

4.3.1. Descripción de los agentes.

El propósito del modelo de agentes es comprender los roles, capacidades y habilidades con las que deben contar los actores dentro de la organización con el fin de lograr sacar provecho al conocimiento que se busca sistematizar. Es común utilizar diagramas de caso de uso para presentar esta información de forma visual ante los stakeholders del proyecto.

En la tabla 9 se tiene el formulario AM-1, en el que recoge de manera concentrada la información relevante acerca de los agentes o actores del sistema.

Tabla 9. Formulario AM-1.

Modelo de Agentes	**Agente AM-1**
Nombre	Nombre del agente

Organización	Posición del agente dentro de la organización y el tipo (humano o software)
Involucrado en	Tareas en las que juega un rol.
Se comunica con	Nombres de los otros agentes con los que interactúa.
Conocimiento	Objetos de conocimiento que posee el agente.
Otras habilidades o capacidades	Habilidades, capacidades requeridas por el agente.
Responsabilidades y restricciones	Responsabilidades del agente en la ejecución de la tarea y sus restricciones. Las restricciones se evalúan en nivel de autoridad que tiene el agente sobre la tarea, así como normal legales internas y externas.

Este documento en conjunto con el TM-1 y el TM-2, contienen la información para el proceso final de la toma de decisiones acerca de nuevos cambios o mejoras en la organización.

4.3.2. Documento de toma de decisiones sobre impactos y mejoras

El documento final de la etapa de modelado contextual en la metodología CommonKADS es el Organization, Task, Agent Models (OTA). En la tabla 10 se describe la información con la que se debe llenar este formulario.

Tabla 10. Formulario OTA-1.

Modelos de Organización, Tareas y Agente.	**Documento de decisión sobre impacto y mejoras OTA-1**
Impacto y cambios en la organización	Comparación de la situación actual de la organización (OM-2) con la situación futura, luego de la implementación del SBC. Se hace para cada uno de los puntos desde una perspectiva global.
Impacto específico y cambios en la tarea/agente.	Cambios o mejoras traídas por el sistema sobre las tareas y los agentes que la ejecutan: 1. Cambios en el flujo, dependencias, objetos manipulados, tiempo y control. 2. Recursos requeridos. 3. Cambio de personal encargado.

	4. Cambios en niveles de autoridad, posiciones, responsabilidades y restricciones en la ejecución de la tarea. 5. Nuevos criterios de rendimiento y calidad. 6. Requerimientos de conocimiento y capacidades en el personal. 7. Cambios en la comunicación.
Actitudes y compromisos.	Reacciones de los actores y stakeholders ante los cambios solicitados.
Acciones propuestas.	Se determinan: 1. Mejoras recomendadas. 2. Medidas para alcanzar esas mejoras. 3. Acción del proyecto que impulsa la solución seleccionada. 4. Resultados, costos y beneficios esperados. 5. Condiciones internas o externas que causarían una reconsideración de la propuesta escogida.

5. MODELADO DEL NIVEL CONCEPTUAL EN COMMONKADS

El análisis realizado mediante los formularios del modelo organizacional propicia una perspectiva más detallada del contexto en el que se planea implementar el sistema. Estudiando los objetos de conocimiento, las tareas y los actores o agentes que las ejecutan. La siguiente etapa busca definir la estructura de la información que será empleada por el sistema, sin exponer la manera en que esta será introducida. En pocas palabras se especifica lo que hará el sistema, pero no la forma de hacerlo.

5.1. Modelo de Conocimiento de la metodología CommonKADS.

Se inicia el proceso de modelado del conocimiento, en el que se detallan aspectos como los tipos de conocimiento, su dominio y el nuevo conocimiento que se generará a partir del existente, todo esto sin caer en el detalle de implementación.

5.1.1. Construcción del modelo de conocimiento de CommonKADS.

Para la elaboración o construcción del modelo del conocimiento, se cuentan con tres fases que se describen a continuación:

5.1.1.1. Identificación del conocimiento

Algo fundamental es verificar las fuentes de procedencia del conocimiento que será insertado en el sistema, en lo cual es recomendable la creación de un glosario en el que se albergue la terminología propia del conocimiento de manera que tanto los ingenieros, expertos y usuarios posean un lenguaje común.

5.1.1.2. Especificación del conocimiento

La especificación del conocimiento se refiere al proceso de clasificación del conocimiento, definiendo las entradas y salidas, y a su vez el conocimiento inferido. Una herramienta muy utilizada para esta labor es un diagrama similar al diagrama de clases de UML, con la excepción de que se omite la descripción de los métodos empleados (figura 33). Este diagrama también ofrece la posibilidad de visualizar las relaciones existentes entre los objetos de conocimiento.

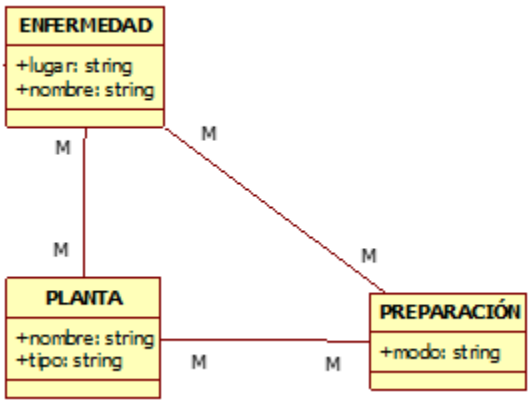


Figura 33. Ejemplo de diagrama de clases para especificación del conocimiento.

5.1.1.3. Refinamiento del conocimiento

Con las relaciones y características del conocimiento definidas, se procede a realizar simulaciones, de las cuales, si se obtienen resultados positivos, es decir, la estructura modelada es una aproximación lo bastante buena al sistema deseado, se da por finalizada la base del conocimiento.

5.1.1.4. Documentación sobre el modelo de conocimiento.

Terminadas las tres fases: identificación, especificación y refinamiento del conocimiento, sólo queda reunirlo todo en un solo formulario, el KM-1 (Henao Cálad, 2001).

Tabla 11. Formulario KM-1.

Modelo de conocimiento	Documentación del conocimiento KM-1
Modelo de conocimientos	Incluye: 1. Diagramas de conceptos. 2. Definición de dominios y subdominios del conocimiento. 3. Tipos de conocimiento (base, inferencia y tarea).
Fuentes estáticas	Listado de textos o fuentes bibliográficas consultadas para la obtención del conocimiento.
Fuentes dinámicas	Personas que, en carácter de experto, aportan su conocimiento al sistema.
Glosario	Terminología propia del conocimiento.

5.1.2. Categorías de conocimiento.

En el formulario KM-1 se hace mención de los tipos de conocimientos sobre el dominio (base), inferencia y tareas. Cada uno de estos conocimientos cuenta con sus características particulares que se detallan en los siguientes puntos:

5.1.1.1. Conocimiento del dominio.

El conocimiento del dominio hace referencia a toda la información que será introducida en el sistema perteneciente a un área específica de estudio, por ejemplo: diagnósticos de enfermedades, en donde se usará una terminología propia de los especialistas en el área entre los que se incluyen nombres de enfermedades, síntomas, entre otros. En otras palabras, es toda la información estática que se posee sobre el área de estudio. Tiene su analogía con el modelo de objetos o modelo de datos en ingeniería de software.

5.1.1.2. Conocimiento sobre inferencias.

Esta categoría del conocimiento recoge lo que vienen siendo los bloques de instrucciones que la máquina utilizará para manipular el conocimiento del dominio, de manera que se genere nuevo conocimiento a partir del existente. Las inferencias son el resultado de la aplicación del conocimiento existente ante nuevos escenarios dentro del mismo dominio.

5.1.1.3. Conocimiento sobre tareas.

El conocimiento sobre las tareas incluye la descripción de las metas u objetivos que debe cumplir dicha tarea y la estrategia para lograrlo. Para esto se requiere la descomposición de las tareas en subtareas y de igual forma añadir una descripción de los controles que se deben considerar durante la ejecución de estas subtareas.

5.1.3. Tipos de tareas proporcionadas por la librería de CommonKADS.

La librería CommonKADS proporciona una clasificación para las tareas que realiza un SBC. Se agrupan en dos tipos: analíticas y sintéticas. La diferencia central entre los dos tipos yace en el tipo de sistema en el que son llevadas a cabo.

5.1.2.1. Tipos de tareas analíticas.

Se reconocen como tareas del tipo analítico aquellas que no requieren de que el sistema se encuentre construido como tal, sino que solo basta con ingresar cierta información sobre el sistema para obtener alguna caracterización del sistema en base a esa información.

Ejemplos de tareas analíticas: clasificación, monitorización, predicción y evaluación.

5.1.2.2. Tipos de tareas sintéticas.

Para el caso de las tareas sintéticas, es imprescindible que el sistema propuesto ya se haya construido, pues las entradas a estas tareas dependen de lo que el sistema les pueda suministrar.

Ejemplos de tareas sintéticas: diseño, modelado, planeamiento, asignación y programación.

5.2. Modelo de Comunicación de la metodología CommonKADS.

El modelo de comunicación es en el que se describen las transacciones o intercambios de datos e información entre las tares realizadas por distintos agentes.

Este proceso de modelado conlleva tres fases al igual que el modelo del conocimiento, y se detallan a continuación:

5.2.2. Plan de comunicaciones.

Es la parte más sencilla del modelo, en este se explican o describen de forma general los agentes que requieren comunicarse durante una serie de tareas dadas y las necesidades que se deben cubrir durante ese intercambio de información.

Para efectos de documentación se procede a realizar un diagrama de diálogo, que consiste en colocar de cada lado a los agentes que intervienen en la tarea, para luego trazar flechas que llevan indicativos del tipo de información que está siendo intercambiada entre los agentes, como se muestra en la figura 34.

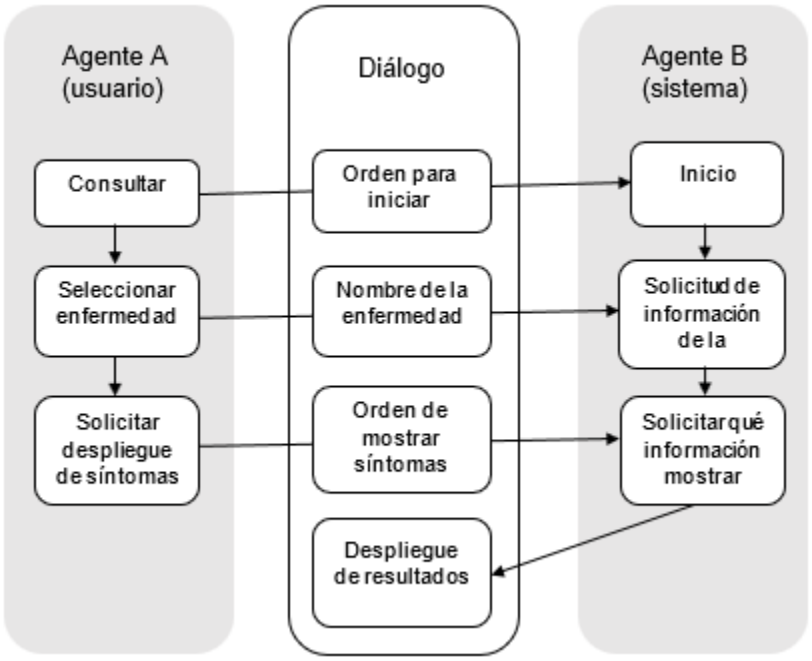


Figura 34. Ejemplo diagrama de diálogo.

5.2.3. Transacciones.

El término transacción dentro del contexto de la metodología CommonKADS, se refiere al intercambio de información entre agentes durante la realización de una tarea dada, en la figura 34, las transacciones quedan dentro del cuadrante de diálogo.

Se aprecia en el ejemplo que las transacciones involucran los tipos de conocimientos como el del dominio (Nombre de la enfermedad) y conocimiento de tareas (órdenes y acciones de despliegue de información). No obstante, no se muestra ningún detalle más allá del nombre de la transacción. Los detalles de la transacción se recogen en un nuevo formulario CM-1, que se aprecia en la tabla 12.

El formulario CM-1 alberga además del nombre de la transacción y los agentes que participan en ella, breves explicaciones relacionadas con el propósito, el contexto y las restricciones existentes para la misma.

Tabla 12. Formulario CM-1.

Modelo de comunicación	Descripción de Transacción CM-1
Nombre o identificador de la transacción	Se requiere la definición de una transacción en cada ocasión que la salida sea un objeto de información. El nombre debe ser significativo, reflejando la acción que realiza con el objeto de información. Se explica el rol de la transacción.
Objeto de información	Se especifica la información que es transmitida y a través de cuáles dos tareas se da esa transmisión.
Agentes involucrados	Especificación del agente que envía y el agente que recibe la información.
Plan de comunicación	Se hace referencia al plan de comunicación al que pertenece esta transacción. Esta descripción cobra mayor importancia en aquellos sistemas en los que se tiene una considerable cantidad de agentes y por consiguiente se tendrá más de un plan de comunicación.
Restricciones	Requerimientos o precondiciones para hacer efectiva la transacción. Si se desea, se añaden las condiciones posteriores en las que quedará el sistema luego de realizada esta transacción para efectos de control de calidad.

5.2.4. Especificaciones del intercambio de información.

Se procede a realizar un refinamiento de la información recabada de la transacción por el formulario CM-1. En la tabla 13, se presenta el formulario CM-2, en el que se añaden elementos de control, sintaxis y otras explicaciones que guardan relación con la estructura interna de la transacción para llegar a las conclusiones que se han descrito en el CM-1.

Tabla 13. Formulario CM-2.

Modelo de comunicación	Especificación del intercambio de información CM-2
Transacción	Nombre o identificador de la transacción.
Agentes involucrados	Agente emisor y receptor.
Objetos de información	Se describen los objetos de información que se intercambian durante la transacción, proporcionando detalles como: si es información esencial o de soporte (auxiliar), el formato y el medio por el cual se transmite.
Especificación de mensajes	Mensajes realizados por la transacción. Para cada mensaje se describe el tipo de comunicación, el contenido y de ser necesario, las referencias.
Control sobre los mensajes	Se realiza una especificación de control sobre los mensajes en caso de ser necesario, mediante el uso de pseudocódigo, en el que se utilizan operadores y estructuras de control básicas.

El modelo de comunicación depende de los modelos de agente, tareas y conocimiento, efectuados en fases anteriores de la metodología CommonKADS, por lo que es recomendable hacer la revisión adecuada de los formularios anteriores de forma previa a la elaboración de este.

6. MODELADO DEL NIVEL COMPUTACIONAL EN COMMONKADS

Luego de efectuado los modelos de conocimiento y comunicación, se pasa a la etapa que corresponde al diseño del sistema, o bien, modelado a nivel computacional. Durante este proceso se pretende mantener la información recolectada sobre el conocimiento a medida que se diseña la arquitectura del sistema, esto implica que el modelo del conocimiento contiene los requerimientos funcionales con los que debe contar el sistema; por otra parte, el modelo de comunicación especifica los requerimientos no funcionales, es decir, todo lo que involucra la interacción con factores externos al sistema. A lo largo del capítulo se describe en qué consiste la estructura del modelo del diseño que sigue la metodología CommonKADS y su desarrollo paso a paso.

6.1. Modelo de Diseño de la Metodología CommonKADS.

Lo primero a destacar del modelo de diseño en CommonKADS es el principio de structure-preserving design (Schreiber et al., 2001) que lo rige. Este principio hace énfasis en no hacer cambios o modificaciones en la información recolectada en los documentos previos a esta fase, pues de lo contrario no se logra mantener la transparencia y se pierde la posibilidad de aplicar métodos de mantenimiento eficientes para el sistema; por otra parte, seguir este principio propicia una alta calidad en el diseño. En el siguiente punto se profundiza un poco sobre el principio de structure-preserving design.

6.1.1. Principio Structure-Preserving Design en CommonKADS.

La idea central de este principio de preservar la estructura del diseño se enfoca en mantener el contenido y estructura de la información del dominio que ha sido presentada en los informes anteriores de forma que este conocimiento pueda ser extraído o recuperado del sistema final, así como las relaciones entre objetos de conocimiento existentes.

Con la idea principal más clara, es preciso agregar que el modelo de diseño incorpora a los estudios anteriores, los detalles relacionados a la implementación

requerida en el software a desarrollar, en términos de mecanismos computacionales y de representación de la información.

Al ser el hecho de conservar la información una prioridad en el proceso de diseño, se cuenta con una serie de criterios que se deben satisfacer:

1. **Reusabilidad del código**: Durante la definición de los fragmentos de código, se enfoca que cada uno corresponda a un rol explícito, obteniendo así una variedad de módulos en términos de tamaño, función y procedimientos de inferencias que en conjunto conforman el sistema final.
2. **Mantenibilidad y adaptabilidad**: La acción de mantener la estructura original de la información relacionada al conocimiento, permite obtener guías o pistas de la estructura del sistema, para así atender posibles omisiones o inconsistencias en el funcionamiento del sistema y corregirlo; dando lugar a posibles extensiones del sistema en un futuro en caso de requerirlo. En pocas palabras, hacer el sistema necesite un mantenimiento más sencillo que los sistemas de información convencionales.
3. **Explicación**: Los procesos de razonamiento detrás de la implementación se explican apoyados del vocabulario dentro del modelo de conocimiento, lo que hace que los bloques de código tengan un nivel de comprensión más elevado al ser revisados, ya que se exponen las situaciones en las que se utilizan y cuál es su papel fundamental.
4. **Apoyo para la obtención de conocimiento**: Puesto que el modelo de conocimiento proporciona elementos descriptivos que contribuyen al entendimiento de los segmentos de código, esta misma información es útil a la hora de obtener, depurar y refinar el sistema.

Este principio es utilizado en la ingeniería de software, sobre todo en las áreas de modelado y diseño orientado a objetos, por el hecho de hacer énfasis en la reusabilidad del código y simplificación del mantenimiento de los sistemas finales (Schreiber et al., 2001).

6.2. Proceso de Diseño en CommonKADS.

Todo proceso de diseño inicia con la especificación de la arquitectura del software de manera general, una vez esto se haya definido se inicia con una especificación más detallada de la arquitectura. En CommonKADS este proceso consta de cuatro fases que se describen a continuación:

6.2.1. Diseñar la arquitectura del sistema.

Dentro de este primer paso hay que tomar en consideración tres lineamientos básicos de cualquier arquitectura:

1. El sistema se descompone en subsistemas.
2. El control general.
3. La descomposición de los subsistemas en módulos del programa.

Para esto, CommonKADS ofrece una guía o plantilla para especificar la arquitectura de cualquier sistema y consta de dos niveles de granularidad:

a. Arquitectura global: Basada en el Modelo Vista Controlador (MVC), se desglosan tres grandes subsistemas:
    1. Modelo de aplicación: Se incluyen las funciones y las bases de conocimiento (estático y dinámico) que darán vida al sistema.
    2. Vistas: Relacionada en mayor parte con las ventanas de la interfaz de usuario, aunque también se colocan las vistas que involucran la interacción del sistema con elementos externos como bases de datos para consultas SQL.
    3. Controlador: Contempla los comandos y controles por los que se rige el sistema, y que manipulan los eventos externos e internos en los cuales se requiere la presencia de un reloj y un proceso de inicio para que el sistema arranque.
b. Arquitectura del subsistema del modelo de aplicación: Se describen los componentes de software que realizan las funciones de razonamiento (tareas e inferencias) y la estructura de la información y el conocimiento.

En la tabla 14 se presenta el formulario DM-1 en el que se resume esta información.

Tabla 14. Formulario DM-1.

Modelo de diseño	Arquitectura del sistema DM-1
Arquitectura de decisión	**Formato.**
Estructura de subsistemas	Aplicación de la variante del modelo MVC.
Modelo de control	Administrador central de los eventos entrantes que utiliza su propio reloj y agenda.
Descomposición de subsistemas	Se descompone el modelo de aplicación en módulos de software, regidos por el principio orientado a objetos.

6.2.2. Identificar la plataforma de implantación.

La plataforma de implantación se refiere al software sobre el que se construirá el sistema y el hardware que lo soportará, es preciso seleccionar cuidadosamente ambos, pues en caso contrario habrá serias afectaciones en los pasos posteriores del diseño. En caso de que el cliente para el que se esté desarrollando no haya impuesto un software específico, existen una serie de lineamientos que se deben tomar en consideración para la selección de una plataforma de desarrollo:

1. Disponibilidad de librerías.
2. Representación declarativa del conocimiento (reglas).
3. Interfaz estándar para la comunicación con otros softwares.
4. Asistente de escritura.
5. Control de flujos.
6. Soporte para CommonKADS.

En la tabla 15 se desglosan estos aspectos con mayor detalle.

Tabla 15. Formulario DM-2.

Modelo de diseño	Plataforma de implantación DM-2
Paquete de Software	Nombre de la plataforma de programación que se utilizará.
Hardware potencial	Listado de posibles candidatos de hardware sobre los que funciona la plataforma de software elegida.

Hardware seleccionado	Hardware que será el responsable de soportar el sistema.
Librerías de visualización	Disponibilidad de librerías para facilitar la construcción de las vistas.
Asistente de escritura del lenguaje	Asistencia al momento de programar, lenguaje orientado a objetos, herencias múltiples, etc.
Representación del conocimiento	Representación declarativa o procedural. Definición de conjuntos de reglas.
Protocolos de interacción	Protocolos soportados para la interacción con agentes externos.
Control de flujos	Mensajes, multihilo, etc.
Soporte CommonKADS	Existencia de herramientas compatibles mediante librerías, que interpreten los modelos de conocimiento y comunicación.

6.2.3. Especificar los componentes de la arquitectura.

Este paso se dedica a ampliar detalles sobre los componentes de la arquitectura, es decir, brindar más información sobre las interfaces que deben existir entre los subsistemas o módulos, sobre todo los aspectos relacionados a las opciones de toma de decisiones creadas por el diseñador y que son aplicables dentro del sistema.

En las plataformas que proveen soporte a la metodología CommonKADS, este paso no requiere de mucho tiempo para ser desarrollado, no obstante, presenta limitaciones en cuanto a creatividad para el diseñador.

En la tabla 16 se describen los elementos que deben ser estudiados o establecidos dentro de la especificación de los componentes de la arquitectura.

Tabla 16. Formulario DM-3.

Modelo de diseño	**Especificación de la arquitectura DM-3**
Componente de la arquitectura	**Puntos de decisión.**
Controlador	Manejador de eventos, determina si un evento proviene de un agente externo o interno. Habilita o deshabilita interrupciones y procesos concurridos. Es una implementación del modelo de comunicación.

Tarea	Para cada tarea se definen las operaciones de inicialización y de ejecución, además de indicar si devuelve algún valor y mensajes de éxito o error del método de tareas.
Método de la tarea	Se define una estructura que cuente con estructuras de IF-THEN y ciclos de repetición. Se debe considerar el tipo de programación, por ejemplo, orientada a objetos; lo que influye en el lugar en donde se declara el método dentro del código.
Inferencia	Basado en la información del modelo de conocimiento. Involucra tres acciones principales: ejecutar, tiene-solución y nueva-solución.
Método de inferencia	Algoritmo(s) utilizados para procesar las inferencias.
Rol dinámico	Toma en consideración los tipos de datos: elemento, conjunto, lista, etc. Y el tipo de modificación que puede sufrir el elemento: selección, añadir, eliminar, vaciar, etc.
Rol estático	Operaciones de acceso: mostrar todos, mostrar uno, verificar existencia de al menos uno, etc.
Base de conocimiento	Representación del conocimiento. Se establece el medio de acceso y modificación a la base.
Vistas	Interfaces gráficas: usuario y experto.

6.2.4. Especificar la aplicación dentro de la arquitectura.

Finalmente, se toman los elementos: tareas, bases de conocimiento, transacciones, entre otros; para ser definidos y ubicados dentro de la arquitectura que se ha diseñado. Para este fin, CommonKADS trabaja con el formulario DM-4 (tabla 17), en el que se resumen las especificaciones finales del modelo del diseño.

Tabla 17. Formulario DM-4.

Modelo de diseño	**Diseño de aplicación DM-4**	
Elemento	**Diseño de decisión**	**Comentarios**
Controlador	Implementación del plan de comunicación y transacciones en control de eventos.	Especifica si el controlador trabaja en tiempo real, concurrencia, etc.

Método de la tarea	Formalización de las estructuras de control.	Utilización de las estructuras de control suministradas por el lenguaje seleccionado.
Rol dinámico	Selección del tipo de dato para cada rol.	Se recomienda la utilización de conjuntos en lugar de listas para una mejor simulación del razonamiento.
Inferencia	Descripción del proceso de invocación de una inferencia.	Demuestra como son utilizados los roles dinámicos y estáticos como argumentos del método de inferencia.
Método de inferencia	Especificación de los métodos de inferencia.	Selección de una técnica o algoritmo de razonamiento apropiado. Se recomienda la reutilización de métodos, en lugar de contar con una cantidad excesiva de estos.
Base de conocimiento	Transcripción del conocimiento mediante un formato de representación provisto por la arquitectura.	Uso de herramientas para representar el conocimiento dentro del sistema.
Vistas	Selección de vistas adecuadas para el modelo de aplicación y los controladores.	En el caso de la vista de usuario, es recomendable utilizar representación del dominio en lenguaje común.

Así termina el estudio, evaluación e implementación de un SBC utilizando la metodología CommonKADS. Todos los formularios son genéricos y no necesariamente deben llenarse en su totalidad, pues no todos los proyectos poseen el mismo nivel de complejidad y hay que adecuarlos de acuerdo con lo que realmente representa aspectos críticos dentro de la organización en la que se desea introducir el SBC.

7. REPRESENTACIÓN DEL CONOCIMIENTO MEDIANTE LÓGICA DE PREDICADOS.

El elemento principal para la construcción de un SBC es el conocimiento, sin embargo, no es posible almacenar este conocimiento en lenguaje común para cualquier persona, debido a que una computadora o sistema informático no se encuentra en la capacidad de procesar este tipo de información, por ende, es necesario recurrir a técnicas o métodos que permiten generar una representación del conocimiento legible y manipulable por las máquinas en las que se pretende implementar el sistema. En el capítulo se exponen las representaciones más comunes: lógica de predicado y lógica proposicional.

7.1. Representación del Conocimiento.

La representación del conocimiento consiste en escribir en un lenguaje que las computadoras puedan procesar, sentencias que describan el conocimiento que se tiene sobre un dominio específico. Los lenguajes empleados para esta tarea cuentan con su propia sintaxis, estructura y reglas.

Esta es la etapa que representa el mayor reto al momento de desarrollar un sistema dentro del ámbito de la inteligencia artificial como lo es un SBC.

7.2. Lógica simbólica o formal.

La lógica simbólica es una ramificación de la lógica que se enfoca en la creación de lenguajes formales, sistemas semánticos y deductivos que ofrezcan la alternativa de efectuar sobre ellos análisis del tipo matemático, de los que generalmente se obtienen teoremas.

Para comprender mejor la lógica simbólica formal, es necesario tener presente el concepto de lenguaje formal. Un lenguaje formal es aquel que cuenta con símbolos, reglas de escritura específicos o sintaxis y una semántica.

La sintaxis se refiere a la forma en que deben estar ordenados los símbolos dentro del lenguaje para formar una sentencia válida; mientras que la semántica verifica

la correspondencia dentro de esa sentencia, garantizando que posea un significado dentro del lenguaje.

Un lenguaje está conformado por un alfabeto (Σ), a partir del cual es posible obtener series de cadenas a las que se les puede aplicar operaciones como concatenación, inversión, longitud, recursividad. De esta manera es válido decir que un lenguaje corresponde al subconjunto de todas las cadenas posibles dentro de un alfabeto (Σ*). Esta terminología suele emplearse sobre todo en la aplicación de autómatas; en donde un autómata corresponde al conjunto de todos los estados, alfabeto de entrada, funciones de transición, estado inicial y estados finales.

Retomando el hilo de la representación simbólica formal o lógica proposicional, es la que cuya unidad más pequeña es la oración o sentencia, de la cual la prioridad no es conocer su significado sino si esta oración en verdadera o falsa (Van Harmelen, Lifschitz, & Porter, 2008). Estas oraciones dentro del contexto de lógica proposicional se denominan proposiciones.

Ejemplos de proposiciones básicas:

Las nubes son blancas o grises.

Luisa es hermana de Enrique.

La lógica proposicional utiliza conectivas para unir las proposiciones básicas. Ejemplos de estas conectivas:

¬ : No, negación.

⋀ : Y

⋁ : O

→: Si, implica.

↔: Si y solo si

Sin embargo, quedan sentencias que quedan fuera del alcance de representación mediante la lógica proposicional, debido a que el contexto que describen no determina un resultado verdadero o falso; además del hecho que una proposición

se trata como un todo, en estos casos, se recurre al uso de la lógica de predicados.

7.3. Representación del conocimiento mediante lógica de predicados.

Mediante la lógica de predicados o lógica de primer orden se pueden representar expresiones más complejas del lenguaje natural, permitiendo así su uso en el proceso de inferencias, sobre todo aquellas que varían de acuerdo al valor que se le otorgue a sus elementos. Esta es la lógica que se aplica con mayor frecuencia en los lenguajes destinados a la creación de SBC (Zeugmann, 2011).

La lógica de predicado cuenta con una serie de elementos que se listan a continuación:

a. Variables proposicionales: Aquellas variables que solo pueden tomar un valor de verdadero o falso. Se denotan con letras minúsculas.
b. Constantes: Se usan para referirse a seres del mundo real o ideal. De modo convencional, utilizan las primeras letras del abecedario.
c. Términos: Corresponden a los sustantivos de la oración y pronombres de la oración. Este concepto incluye a los dos anteriores: variables y constantes.
d. Cuantificadores: Encargado de definir para cuantos con la que un predicado es verdadero. Corresponden a las palabras “todo”, “algunos”, “nunca” y demás expresiones similares.
e. Predicados: Siguiendo el concepto conocido en español, corresponde a las afirmaciones sobre propiedades o relaciones existentes de los términos.

Cada predicado posee una cardinalidad que denota la cantidad argumentos que lo conforman. Un predicado P se interpreta como un conjunto de objetos A que tienen cierta propiedad P, lo que de forma simbólica se expresa: $\{a \in A \mid P(x)\}$. De esta forma, un predicado es asignable a varios términos.

7.4. Inferencia y razonamiento.

La inferencia y razonamiento parten de un elemento principal para todo SBC, la base de conocimiento. Esta base de conocimiento se encuentra conformada por hechos y reglas o sentencias escritas en lenguaje formal, comprensible a nivel computacional.

El propósito fundamental es la obtención de nuevo conocimiento, escrito con la misma sintaxis que el albergado dentro de la base de conocimiento; sin embargo, este nuevo conocimiento es el resultado de un análisis profundo e interpretación del conocimiento previo; se les conoce con el nombre de conclusiones.

Estas conclusiones sólo son ciertas, si y sólo si, se cumplen dentro de todos los casos posibles en lo que se vea involucrada.

7.5. Métodos básicos de razonamiento.

Para obtener el nuevo conocimiento existen dos métodos básicos del razonamiento:

a. Razonamiento hacia adelante: Partiendo de hechos, se le proporcionan una serie de parámetros, que van siendo evaluados para llegar a una conclusión o nuevo hecho (McLeod & Schell, 2001).

Ejemplo:

Hecho: La **jirafa** es un **animal** que tiene el **cuello largo**.

Si: **x** es **animal** y **x** tiene **cuello largo**

Entonces: **x** es una **jirafa**.

b. Razonamiento hacia atrás: Conociendo la solución, recorre las condiciones que deben cumplirse para llegar a ella (McLeod & Schell, 2001).

Ejemplo:

Si: **x** tiene **pico** y **x** tiene **plumas.**

Entonces: **x** es un **ave**.

Los ejemplos presentados son con el fin de observar la diferencia entre los dos tipos de razonamiento descritos, para que una computadora sea capaz de realizar estas inferencias es completamente necesario que tanto los hechos y reglas estén representados a través de un lenguaje formal.

7.6. Reglas para transformar expresiones en cláusulas lógicas.

Una cláusula lógica está conformada por un conjunto finito de literales, donde los literales corresponde a fórmulas atómicas o declaraciones combinadas por conectivas lógicas, como las empleadas en la lógica proposicional (Chávez, Riscos, & Graciani, 2012).

A continuación, se desarrolla un ejemplo de transformación de una expresión a una cláusula lógica.

Teniendo el enunciado: **José** es el padre de **Ricardo**.

Se identifican los términos dentro del enunciado: José y Ricardo.

Se obtiene el predicado: es el padre de.

Siguiendo lo dicho en la sección 7.3. punto C, los términos se proceden a reemplazar con letras, por ejemplo, m para José y n para Ricardo. De igual manera, el predicado o relación entre ambos se reemplaza con una letra mayúscula, en este caso P.

La expresión queda de la siguiente forma P(m,n) que se lee “m es padre de n”; de lo que se asume que “n no es padre de m” o bien, ¬P(n,m). Colocando las dos expresiones juntas, queda de la siguiente forma.

$$P(m,n) \rightarrow \neg P(n,m)$$

Es válido dejar la expresión “Padre” como predicado en lugar de la P, no obstante, en las situaciones en donde se requiera hacer uso de este predicado una cantidad considerable de veces resulta más cómodo utilizar una representación.

7.7. Ejemplos de representación de hechos.

Para representar hechos, la lógica de predicado se vale de otros símbolos como:

∀: Que significa “para todo”.

∃: Que denota que “existe al menos uno”.

| : Se lee “tal que”

Estos símbolos en combinación con las conectivas vistas de la lógica proposicional son las herramientas utilizadas en la lógica de predicado para representar hechos.

Visto el proceso de conversión de una expresión a una cláusula lógica, se procede a convertir los siguientes hechos:

1. Si no llueve mañana, Luis irá a la playa.

¬Clima(mañana, lluvia) → Ir(Luis, playa)

2. Todos los participantes son hombres.

∀X Participante(X, hombre)

3. Algunos estudiantes juegan ajedrez.

∃X Juega(X, ajedrez)

4. Todas las personas que juegan baloncesto son altas.

∀X∀Y (personas(X) Λ juegan(X,Y) Λ baloncesto(Y)) ⟶ altas(X).

5. Alguna persona le gusta los camarones

∃X (persona(X) Λ gusta(X, camarones)).

6. A nadie le gustan los impuestos.

¬∃X gusta(X, impuestos) ò ∀X ¬ gusta(X, impuestos)

El proceso convertir una representación a lenguaje natural es más sencillo que lo inverso, debido a las múltiples lógicas que pueden ser aplicables sobre los mismos hechos.

BIBLIOGRAFÍA

Acronymics Inc. (2004). Agentbuilder: An Integrated Toolkit for Constructing Intelligent Software Agents.

Amatriain, H. (2015). Sistemas Expertos Y Sistemas Basados En Conocimientos. *Universidad Nacional De Lanus*, 13.

Badaró, S., Ibañez, L. J., & Agüero, M. J. (2013). Sistemas Expertos: Fundamentos, Metodologías y Aplicaciones. *Revista de Ciencia y Tecnología*, *13*, 349–363. https://doi.org/http://dx.doi.org/10.18682/cyt.v1i13.122

Carlos Soto, M. (2002). *Sistema experto de diagnostico medico del Síndrome de Guillian Barre.* Retrieved from http://sisbib.unmsm.edu.pe/bibvirtualdata/Tesis/Basic/carlos_sm/cap2.pdf

Chávez, A., Riscos, A., & Graciani, C. (2012). Lógica y Programación. Cláusulas y formas clausales.

Clocksin, W. F., & Mellish, C. S. (2012). *Programming in Prolog: Using the ISO Standard*. Springer Berlin Heidelberg. Retrieved from https://books.google.es/books?id=wuERBwAAQBAJ

Exsys Corvid. (2011). Corvid - Quick Start Guide.

Friedman-Hill, E. (2003). Jess, the expert system shell for the java platform, (November 1997). Retrieved from http://www.iau.dtu.dk/teaching/31380/Jess/manual.pdf

Garcia, M., & Rodríguez, J. (2013). Sistemas Basados En El Conocimiento. *Vínculos*.

Giarratano, J. C. (2017). CLIPS 6.4 User's Guide.

González, I. (2007). Programación Lógica : Introducción y Sintaxis Básica.

Harmon, P., & King, D. (1988). *Sistemas expertos: aplicaciones de la inteligencia artificial en la actividad empresarial*. Díaz de Santos.

Henao Cálad, M. (2001). *CommonKADS-RT: Una Metodología para el Desarrollo de Sistemas Basados en el Conocimiento de Tiempo Real*. Retrieved from http://users.dsic.upv.es/grupos/ia/sma/thesis/pdf/TesisMonicaH.pdf

McLeod, R., & Schell, G. (2001). Management Information Systems (pp. 13–51). https://doi.org/10.1017/CBO9781107415324.004

Noyes, J. L. (1992). *Artificial Intelligence with Common Lisp: Fundamentals of Symbolic and Numeric Processing*. D.C. Heath. Retrieved from https://books.google.com.pa/books?id=eIbBm7wvTjcC

Pino Díez, R., Gómez Gómez, A., & de Abajo Martínez, N. (2001). *Introducción a la inteligencia artificial: sistemas expertos, redes neuronales artificiales y computación evolutiva*. Servicio de Publicaciones, Universidad de Oviedo.

Riley, G. (2017). CLIPS Reference manual Interfaces Guide.

Russell, S., & Norvig, P. (2004). *Inteligencia Artificial: un enfoque moderno* (Segunda Ed). Madrid: Pearson Education.

Sajja, P. S., & Akerkar, R. (2010). Knowledge-Based Systems for Development. *Advanced Knowledge Based Systems: Model, Applications & Research*, *1*, 1–11. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Success+Factors+in+Implementing+Knowledge+Based+Systems#0

Schreiber, G., Akkermans, H., Anjewierden, A., Hoog, R., Shadbolt, N., Van Velde, W., & Wielinga, B. (2001). *Knowledge Engineering and Management - The CommonKADS Methodology* (Vol. 24).

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge Engineering: Principles and methods. *Data & Knowledge Engineering*, *25*, 161–197. https://doi.org/10.1016/S0169-023X(97)00056-6

Van Harmelen, F., Lifschitz, V., & Porter, B. (2008). *Handbook of Knowledge Representation*.

Zeugmann, T. (2011). Predicate Logic, 782–782. https://doi.org/10.3109/14767058.2013.799660

ANEXOS

Universidad Tecnológica de Panamá

Facultad de Ingeniería de Sistemas Computacionales

Sistemas Basados en el Conocimiento

Prueba #1

Nombre: _____ Cédula: _____ Grupo: _____

Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____ /

I Parte. Llene los espacios.

1. Tres estrategias para la representación del conocimiento son: _____, _____ y _____.

2. Los cuatro tipos de SBC son basados en: _____, _____, _____ y _____.

3. Componente de un SBC que almacena el conocimiento de los hechos y la heurística: _____.

4. El conocimiento _____ es aquel que se refiere al conocimiento de las buenas prácticas y el buen juicio.

5. El conocimiento _____ es aquel que es ampliamente compartido y que puede encontrarse en libros.

6. Componente de un SBC que muestra el contenido de las reglas almacenadas en la base de conocimiento de modo que sea entendible para cualquier persona: _____.

II Parte. Desarrollo.

1. ¿Qué son los sistemas basados en el conocimiento?

2. Describa una de las estrategias utilizadas por el motor de inferencias.

Universidad Tecnológica de Panamá

Facultad de Ingeniería de Sistemas Computacionales

Sistemas Basados en el Conocimiento

Prueba #2

Nombre: _____ Cédula: _____ Grupo: _____

Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____ /

I Parte. Llene los espacios.

1. Mencione dos lenguajes que pueden utilizarse en la programación de un SBC: _____ y _____.

2. Mencione cuatro aplicaciones de un SBC: _____, _____, _____ y _____.

3. Dos ejemplos de shells son: _____ y _____.

4. Un aspecto que el ingeniero del conocimiento debe tomar en consideración al identificar el problema que se busca resolver con un SBC es: _____.

5. Mencione una razón para utilizar un SBC: _____.

II Parte. Desarrollo.

1. Mencione las cinco etapas de desarrollo de un SBC.

2. Indique una ventaja y una limitación del uso de un SBC.

Universidad Tecnológica de Panamá

Facultad de Ingeniería de Sistemas Computacionales

Sistemas Basados en el Conocimiento

Prueba #3

Nombre: _____ Cédula: _____ Grupo: _____

Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____ /

I Parte. Cierto (C) y falso (F).

1. ____ Los tres tipos de nodos con los que trabajan los árboles de decisión en ES-Builder son Attributes, Values y Conclusion.
2. ____ En Exsys Corvid, las reglas se agrupan en bloques de comandos.
3. ____ En Prolog, la palabra dominio se refiere a los tipos de datos de las variables que serán utilizadas por el programa.
4. ____ Dos maneras de representar el conocimiento dentro de CLIPS son las reglas y las funciones.
5. ____ La herramienta JESS es una variante de CLIPS escrita puramente en Java.
6. ____ CLIPS significa C Language Integrated Production System.
7. ____ En Prolog, las directivas de compilación se refieren a las características del compilador.
8. ____ En Exsys Corvid, los bloques de lógicos le indican al sistema qué hacer.

II Parte. Desarrollo.

1. Mencione cuatro tipos de datos dentro de la herramienta JESS.

2. Mencione los tres componentes principales de un programa básico en CLIPS.

Universidad Tecnológica de Panamá

Facultad de Ingeniería de Sistemas Computacionales

Sistemas Basados en el Conocimiento

Prueba #4

Nombre: _____ Cédula: _____ Grupo: _____

Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____ /

I Parte. Cierto (C) y falso (F).

1. ____ La herramienta ES-Builder ordena el conocimiento siguiendo una estructura de árbol de decisión.
2. ____ La principal característica de los SBC basados en lógica difusa es la incertidumbre.
3. ____ CLIPS es una versión del lenguaje LISP.
4. ____ La herramienta AgentBuilder está basada en C.
5. ____ Un SBC puede mantener una conversación en lenguaje natural.
6. ____ Los SBC basados en redes bayesianas emplean probabilidades.
7. ____ El módulo de explicación de un SBC provee información adicional acerca de los resultados arrojados por el sistema.
8. ____ La herramienta Exsys Corvid requiere de conocimientos de programación para la creación de un sistema experto.
9. ____ El motor de inferencias emplea estructuras de control para efectuar el razonamiento.

II Parte. Desarrollo.

1. Indique las seis secciones que conforman la estructura de los programas escritos en Prolog.

Universidad Tecnológica de Panamá

Facultad de Ingeniería de Sistemas Computacionales

Sistemas Basados en el Conocimiento

Prueba #5

Nombre: _____ Cédula: _____ Grupo: _____

Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____ /

I Parte. Llene los espacios.

1. En el nivel computacional de la metodología CommonKADS, se encuentra el siguiente modelo: _____.

2. En el nivel contextual de la metodología CommonKADS, se encuentran los siguientes modelos: _____, _____ y _____.

3. En el nivel conceptual de la metodología CommonKADS, se encuentran los siguientes modelos: _____ y _____.

4. Originalmente, CommonKADS era llamado: _____.

II Parte. Desarrollo.

1. Mencione las seis etapas que forman parte del proceso de producción de un SBC.

2. ¿A qué se refiere el término metodología?

Universidad Tecnológica de Panamá

Facultad de Ingeniería de Sistemas Computacionales

Sistemas Basados en el Conocimiento

Prueba #6

Nombre: _____ Cédula: _____ Grupo: _____

Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____ /

I Parte. Llene los espacios.

1. El modelado a nivel contextual en CommonKADS se subdivide en: _____, _____ y _____.

2. En el modelo de agentes, para presentar información de forma visual ante los stakeholders es común utilizar: _____.

3. Dentro del modelo organizacional, dos formas de obtener información para la descripción del problema son: _____ y _____.

4. Dos áreas que se evalúan en el análisis de viabilidad, dentro del modelo organizacional son: _____ y _____.

5. Mencione dos categorías en la que se dividen los stakeholders para un proyecto de SBC: _____ y _____.

6. El documento final de la etapa de modelado contextual en CommonKADS se denomina: _____.

II Parte. Desarrollo.

1. Mencione dos características de una tarea, dentro del modelo de tareas en CommonKADS.

2. ¿Cuál es el propósito del modelo de agentes?

Universidad Tecnológica de Panamá

Facultad de Ingeniería de Sistemas Computacionales

Sistemas Basados en el Conocimiento

Prueba #7

Nombre: _____ Cédula: _____ Grupo: _____

Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____ /

I Parte. Llene los espacios.

1. Nombre del proceso de clasificación del conocimiento: _____.

2. Mencione dos ejemplos de tareas analíticas: _____ y _____.

3. Mencione tres ejemplos de tareas sintéticas: _____, _____ y _____.

4. Mencione las tres categorías del conocimiento, dentro del modelo de conocimiento: _____, _____ y _____.

II Parte. Desarrollo.

1. Mencione las tres fases del proceso de modelado de comunicación.

2. Mencione las tres fases principales para la construcción del modelo de conocimiento de CommonKADS.

Universidad Tecnológica de Panamá

Facultad de Ingeniería de Sistemas Computacionales

Sistemas Basados en el Conocimiento

Prueba #8

Nombre: _____ Cédula: _____ Grupo: _____

Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____ /

I Parte. Llene los espacios.

1. Mencione dos lineamientos básicos que se deben tomar en cuenta durante el diseño de la arquitectura del sistema: _____ y _____.

2. Mencione cuatro criterios que deben cumplirse según el principio Structure-Preserving Design en CommonKADS: _____, _____, _____ y _____.

3. Nivel de arquitectura del sistema que se basa en el Modelo Vista Controlador (MVC): _____.

4. Mencione cuatro lineamientos que se deben considerar al seleccionar una plataforma de desarrollo: _____, _____, _____ y _____.

II Parte. Desarrollo.

1. Mencione las cuatro fases del proceso de diseño en CommonKADS.

Universidad Tecnológica de Panamá

Facultad de Ingeniería de Sistemas Computacionales

Sistemas Basados en el Conocimiento

Prueba #9

Nombre: _____ Cédula: _____ Grupo: _____

Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____ /

I Parte. Llene los espacios.

1. Verifica la correspondencia dentro de una sentencia válida, garantizando que posea un significado dentro del lenguaje: _____.

2. Se refiere a la forma en que deben estar ordenados los símbolos dentro del lenguaje para formar una sentencia válida: _____.

3. Tres operaciones que se le pueden a aplicar a una serie de cadenas en un lenguaje son: _____, _____ y _____.

4. Los métodos básicos de razonamiento son: _____ y _____.

5. Todo lenguaje formal cuenta con los siguientes elementos: _____, _____ y _____.

II Parte. Desarrollo.

1. Indique el símbolo y significado de cinco conectivas utilizadas en la lógica proposicional. 5

Universidad Tecnológica de Panamá

Facultad de Ingeniería de Sistemas Computacionales

Sistemas Basados en el Conocimiento

Prueba #10

Nombre: _____ Cédula: _____ Grupo: _____

Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____ /

I Parte. Cierto (C) y falso (F).

1. ____ Los dos métodos básicos de razonamiento son el razonamiento hacia adelante y el razonamiento hacia atrás.
2. ____ La semántica se refiere a la forma en que se deben ordenar los símbolos dentro del lenguaje para formar una sentencia válida.
3. ____ Un lenguaje es el subconjunto de todas las cadenas posibles dentro de un alfabeto.
4. ____ Un predicado es una afirmación sobre la propiedad de un objeto.
5. ____ En el razonamiento hacia atrás, se parte de los hechos.
6. ____ Un cuantificador define para cuántos objetos un predicado es verdadero.
7. ____ Un lenguaje formal únicamente cuenta con sintaxis y semántica.
8. ____ Un alfabeto se representa con el símbolo Σ*.
9. ____ En la lógica proposicional, la unidad más pequeña es la oración.
10. ____ Un autómata corresponde al conjunto de todos los estados, alfabeto de entrada, funciones de transición, estado inicial y estados finales.

II Parte. Desarrollo.

1.Mencione los cinco elementos que forman parte de la lógica de predicado.

1

CAPÍTULO I: CONCEPTOS, CARACTERÍSTICAS Y ESTRUCTURA BÁSICA DE LOS SBC

2

CONCEPTOS, CARACTERÍSTICAS Y ESTRUCTURA BÁSICA DE LOS SBC

Los avances tecnológicos que han surgido con el pasar de las décadas, han dejado marcadas las diferencias entre los sistemas creados dentro del margen de la inteligencia artificial, en los que se consideran aspectos como el tipo de procesamiento que es aplicado sobre los datos con los que se alimentan dichos sistemas, tal es el caso de los sistemas basados en el conocimiento (SBC). Este capítulo se enfoca en definir los conceptos fundamentales, características y estructuras que poseen los SBC.

3

Definición de los SBC

Los sistemas basados en el conocimiento (SBC) son una clase de sistemas capaces de albergar una representación del conocimiento de un área de aplicación en específico, estando en la posibilidad de brindar soluciones óptimas frente a una problemática dada, mediante el uso de métodos de razonamiento elaborados para sacar el mayor provecho del conocimiento almacenado y poder transmitirlo de manera similar a la que lo haría un experto. A menudo se suele referir a los SBC como sistemas expertos.

4

Características de los SBC

- Se especializa en un área de conocimiento específica.
- Resuelve problemas con un grado de dificultad elevado, propios de un dominio determinado.
- Posee la capacidad de manejar grandes volúmenes de información.
- Utilizan razonamiento simbólico.
- Aplica nuevas estructuras y modos de organización del conocimiento.
- Aplica el conocimiento que posee para realizar inferencias a partir de nuevos datos.
- Manejan incertidumbre.
- No usan procedimientos algorítmicos.
- Procuran obtener las soluciones óptimas.
- Emplean datos mayormente cualitativos.
- Pueden comunicarse con expertos para nutrir su base de conocimientos.
- Justifican los resultados obtenidos en caso de ser solicitado por el experto o usuario.
- Debe estar en capacidad de manipular, modificar, actualizar y ampliar datos.

5

Tipos de SBC

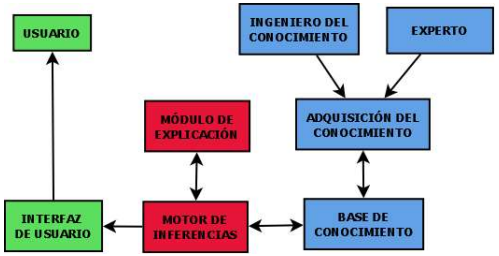
Los SBC varían en la manera en que son diseñados y construidos, estas diferencias radican con mayor incidencia en el método escogido para almacenar, procesar e inferir el nuevo conocimiento. De esta forma, los SBC se clasifican considerando su aplicación o técnica de razonamiento. Dentro del marco de modo de representación y manipulación del conocimiento para la generación de respuestas o soluciones, se tienen los siguientes tipos de SBC:

- **Basados en reglas:** Trabaja con hechos y reglas, siendo características estructuras de control como IF-THEN, etiquetados, entre otros; mediante los que se van comparando los resultados obtenidos dadas determinadas condiciones iniciales.
- **Basados en casos**: Su mayor distinción está en el uso de la experiencia obtenida para la resolución de un problema para solventar situaciones futuras. Este tipo de SBC hace una analogía con la forma en la que el ser humano tiende a razonar en su diario vivir.
- **Basados en redes bayesianas**: Emplea probabilidades para determinar las relaciones entre un conjunto de variables dentro de un modelo de grafo acíclico dirigido.
- **Basados en lógica difusa**: La incertidumbre es su característica principal; trata de simular el razonamiento humano, que tiende a tener más opciones que un simple Sí o No. Esta incertidumbre es aplicada mediante conjuntos difusos, que causan que el sistema trabaje de forma menos precisa, pero con una mayor lógica "humana".

6

Componentes de los SBC

Los SBC poseen una estructura básica en la que se distinguen una serie de componentes cuyas relaciones se pueden apreciar en la figura.



7

Base del conocimiento

En ella se almacena el conocimiento de los hechos y la heurística, factores necesarios para comprender, formular y ofrecer una solución a los problemas propuestos. Se alimenta mediante el módulo de adquisición de conocimiento. Se distinguen dos tipos de conocimiento:

- Factual: Es aquél que es ampliamente compartido y que puede encontrarse en revistas o libros, corresponde al dominio de la aplicación.
- Heurístico: Se refiere al conocimiento de las buenas prácticas, buen juicio y el razonamiento plausible en el campo. Este tipo de conocimiento no es fácil de representar, dado su grado de subjetividad e incertidumbre.

Entre las estrategias para representar el conocimiento están:

- Autómatas finitos: Descompone el conocimiento en fragmentos estáticos, es decir, difíciles de alterar o cambiar.
- Cálculos de predicados: Emplea lógica de proposiciones.
- Reglas de producción: Hace uso de condiciones y comparativas.
- Redes semánticas: Utiliza grafos en donde los nodos son conceptos y los arcos, relaciones.
- Objetos estructurados: Combina las anteriores para obtener una representación más acertada.

8

Motor de inferencias

Actúa como el cerebro del sistema, en el que se emplean estructuras de control y una metodología para efectuar el razonamiento. Este camino puede ser encontrado utilizando dos tipos de estrategias:

- Encadenamiento hacia adelante: Aplicable cuando se tiene un número de resultados muy amplio por lo que es necesario construir la solución, acción realizada evaluando los hechos conocidos contra las reglas aplicables en la base de conocimiento.
- Encadenamiento hacia atrás: Mayormente empleado cuando la cantidad de resultados posibles es conocido y bastante reducido; en donde el proceso realizado conlleva determinar qué reglas pueden llevar al objetivo o solución que se espera, es decir, que se van desglosando todas las condiciones que son necesarias para que ocurra ese objetivo a medida que se van encontrando reglas aplicables.

9

Módulo de adquisición del conocimiento

El módulo de adquisición de conocimiento es el encargado de capturar el conocimiento, ya sea de los expertos o el que está documentado en las fuentes bibliográficas que se emplean para la construcción del sistema experto.

Su misión principal es ayudar a construir las bases del conocimiento permitiendo su posterior ampliación o actualización.

Entre las técnicas empleadas para la adquisición del conocimiento se encuentran los análisis de protocolo, entrevistas y observación.

10

Módulo de explicación

Es una unidad creada con el fin de poder ofrecer información adicional acerca de los resultados arrojados por el sistema.

Las preguntas o solicitudes de las razones de los resultados pueden ser realizadas por los usuarios.

Esta función contribuye con una de las misiones de los SBC que es la transmisión del conocimiento como si tratase de un tutor.

11

Interfaz de usuario

Como toda herramienta de software que se ofrecerá a un público determinado, es necesario que cuente con una interfaz lo suficientemente agradable y comprensible por los mismos, en los que se proporcione menús y ventanas de diálogos que faciliten su uso.

En la interfaz se muestra el contenido de las reglas almacenadas en la base de conocimiento de forma entendible para cualquier persona.

12

Desarrollo de un SBC

El recurso fundamental para iniciar un proyecto en el que se busca construir un SBC es el conocimiento en sí, debido a que este es la razón de existir del sistema.

El motivo principal por el que se implementa un SBC es la necesidad de transmitir y ofrecer acceso permanente al conocimiento de un experto, a cualquier persona indistintamente de su nivel de educación, de manera que este pueda ser aprovechado en su totalidad.

13

Etapas de desarrollo de un SBC

El desarrollo de un SBC, como cualquier otro tipo de programa convencional, tiene una serie de etapas o fases para su construcción desde cero. Distinguimos cinco etapas, cada una con sus características:

1. Identificación del problema

Siendo el primer paso en que se involucra un alto nivel de análisis y estudios de factibilidad, se debe tener un especialista en el área en el que se quiere aplicar el SBC y un ingeniero del conocimiento.

14

Modo de desarrollo

Un recurso indispensable para poder adentrarse en esta fase, son los requerimientos del SBC, discutidos en la fase anterior.

En base a los requerimientos, el ingeniero del conocimiento puede proceder a la selección de las herramientas necesarias para la obtención de las características deseadas en el sistema.

Elección de lenguajes, herramientas y conchas

- Entre los lenguajes creados para la programación de SBC se tienen: PROLOG, LISP
- De igual forma, se tienen Shells o conchas como: Java Expert System Shell (JESS), Jena, JEOps

Desarrollo de un prototipo

- Se lanzan las primeras versiones del sistema, en las que se realiza un proceso de pruebas y rectificación sobre la base de conocimiento que ha sido construida, tal cual se realizaría para un desarrollo de software convencional, en el que se ejecutan validaciones y verificaciones.

Planificación de un sistema a gran escala

- Es común que, dado el éxito de la implementación de un sistema pequeño, se busque la manera de incrementar su tamaño y potencia, este motivo es por el cual precisamente la elección adecuada de las herramientas

Implementación, mantenimiento y evolución

- En esta fase el sistema es colocado en un entorno en donde entrará en contacto directo con sus usuarios finales, quienes no necesariamente fueron los que impulsaron el desarrollo del SBC.

15

Aplicaciones de un SBC

Un SBC es creado con el propósito de ser aplicado en un área o tarea determinada, de modo que pueda servir como guía o consulta dependiendo del nivel de conocimiento de su usuario. Teniendo esto en consideración, se puede subdividir a los SBC de acuerdo con las funciones que realicen:

- Clasificación: Diseñados para Identificar un objeto en base a las características que presenten.
- Sistemas de diagnóstico: Dados una serie de datos observables, proporcionados por el usuario, infiere qué problema, malfuncionamiento o enfermedad se está presentando.
- Monitoreo: Prescribe el comportamiento de un sistema, mediante la comparación continua de los datos generados por el mismo en el tiempo.
- Control de procesos: Mediante un seguimiento se verifica la correcta ejecución de las tareas o procesos.
- Diseño: Orientado a generar un resultado basado en las especificaciones indicadas por el usuario.
- Planificación: Desarrollar o modificar un plan de acción tomando en cuenta determinados parámetros de eficiencia.
- Generación de opciones: Brinda soluciones o alternativas a un problema, cada una con un grado menor o mayor de eficacia.

16

Razones para utilizar un SBC

Luego de haber visto los conceptos fundamentales y el ciclo de vida de los SBC, hay que destacar las situaciones en las que es realmente válida la implementación de un sistema de este tipo:

- No hay suficientes expertos humanos en un área del conocimiento.
- Cuando el conocimiento debe ponerse a disposición constante.
- Situaciones en los que la subjetividad humana, pueda acarrear soluciones equivocadas.
- El volumen de datos a procesar es muy elevado para obtener un resultado.
- Cuando el conocimiento de más de un experto debe ser almacenado en una sola plataforma.

17

Ventajas y Limitaciones

Entre las ventajas más sobresalientes de la aplicación de un SBC para la solución de problemas están:

- Mayor rapidez en la obtención de respuestas.
- Reducción del error humano.
- Disminuyen la inversión de salarios de recursos humanos.
- Colocan a disposición de cualquiera el conocimiento que almacenan.
- Pueden estar en servicio en cualquier momento.
- Capacidad para trabajar con información parcial como un experto humano.
- Aprende de su experiencia previa.

Por otra parte, algunas de las limitaciones detectadas de forma general para el uso de un SBC, son las siguientes:

- Carecen de sentido común.
- No pueden mantener una conversación en lenguaje natural.
- Para que identifique errores es necesario intervenir a nivel de programación.
- Requiere que se ingrese conocimiento estructurado.

18

CAPÍTULO II: HERRAMIENTAS PARA EL DESARROLLO DE SBC

19

HERRAMIENTAS PARA EL DESARROLLO DE SBC

Para el desarrollo de un SBC se puede hacer uso de cualquier lenguaje de programación de uso genérico, sin embargo, el proceso puede tornarse más lento y extenso dado que se requiere diseñar y codificar el sistema desde cero; no obstante, existe una gama de software creados con el fin de crear sistemas de este tipo, que incluyen sus propias estructuras para la representación del conocimiento, así como sus propios motores de búsqueda e inferencias.

Este capitulo se dedica a repasar el uso básico de algunas de las herramientas para el desarrollo de SBC que fueron vistas en cursos previos como Inteligencia Artificial y Herramientas Aplicadas a la Inteligencia Artificial.

20

Expert System Builder

Mejor conocido como ES-Builder, es una herramienta gratuita que permite desarrollar las bases de las habilidades para la construcción de un SBC, pues ofrece una interfaz en la que se puede observar cómo es ordenado el conocimiento siguiendo una estructura de árbol de decisión.

Desarrollado por McGoo Software, con fines educativos, en un principio contaba con dos versiones: la versión de escritorio para Windows y la versión web; sin embargo, actualmente solo se encuentra disponible la versión web.

21

Nodos

ES-Builder, al trabajar con árboles de decisión, utiliza nodos, específicamente tres tipos de nodos que responden a una jerarquía dentro del árbol:

Attributes: El nodo *Attribute* o atributo, se encuentra en el segundo nivel del árbol. Este nodo está representado mediante una letra 'A' en color azul.

Values: En el caso del nodo *Value* o valor, se encuentra justo en el siguiente nivel que un nodo de atributo. Es representado con la letra 'V' en tonalidad rojo vino.

Conclusión: El nodo *Conclusion* o conclusión, por su parte, cumple el papel de hoja en el árbol, ya que una conclusión es el nodo final de una rama. Su representación es una letra 'C' en color verde.

22

ExSys Corvid

Exsys Corvid es una poderosa herramienta, de propósito general, con un ambiente de desarrollo intuitivo, en el que los expertos pueden describir el conocimiento mediante un conjunto de reglas en el idioma inglés y un poco de algebra.

Esta herramienta ha sido diseñada con el fin de no requerir conocimientos de programación para su utilización en el proceso de creación de sistemas expertos.

23

Variables

Las variables dentro de Corvid poseen similitud con las variables de cualquier otro lenguaje de programación, por lo tanto, son elementos nombrados por el programador y están asociados a un tipo de valor. Las variables son un elemento fundamental para el diseño de un sistema, pues pueden adoptar el valor que el usuario indique, ser probadas y modificadas por las reglas que rigen al sistema inteligente, debido a esto, se deben seguir ciertas pautas al momento de su definición:

- Name (Nombre): Debe ser único y no debe contener caracteres especiales. No existe límite de longitud; no obstante, se recomienda que sea lo más específico y corto posible pero que refleje su papel dentro del sistema.
- Prompt (Mensaje, aviso): Es el enunciado o pregunta que se va a mostrar al usuario. Debe estar relacionado con el tipo de valor que se espera almacenar en la variable.

24

Creación de Bloques Lógicos

Las reglas en Exsys Corvid son agrupadas dentro de los llamados bloques lógicos. Estos bloques pueden contener desde una simple regla hasta varias estructuras de árboles de reglas con la única condición de que estas se encuentren relacionadas entre sí para la resolución de un aspecto en la toma de decisiones.

25

Bloques de Comandos

Los bloques lógicos determinan la manera cómo realizar las cosas, mientras que los bloques de comandos le indican al sistema qué hacer. Un bloque de comandos le dice al motor de inferencia de Corvid la manera en que se deben aplicar las reglas. Un bloque de comando realiza cualquier tipo de acción en el sistema, sin afectar la lógica de este.

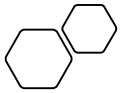
- Entre los usos comunes para los bloques de comando están los siguientes:
- Determinar el enfoque de encadenamiento, hacia adelante o hacia atrás.
- Controlar el orden en que se ejecutan los bloques lógicos.
- Obtener datos de recursos externos al sistema.
- Transmitir los resultados y recomendaciones a otros programas.

26

Ejemplo: Solicitud de Préstamo

- Paso 1: Crear el proyecto
- Paso 2: Declarar las variables y asignar sus valores
- Paso 3: Añadir los bloques lógicos (reglas)
- Paso 4: Definir los bloques de comando básicos
- Paso 5: Ejecutar el sistema

27



Prolog

Prolog es un lenguaje de programación del tipo descriptivo que está enfocado en la declaración de objetos y las relaciones que hacen ciertas una solución para un problema, lo que quiere decir que su objetivo es la descripción del conocimiento que se tiene sobre el problema y no la secuencia de pasos necesarios para resolverlo.

28

Directivas de compilación

Las directivas de compilación se refieren a las características del compilador. Son opciones que pueden modificarse desde el menú, líneas de comando o dentro del propio código fuente del programa que se esté trabajando. Entre las características modificables del compilador se encuentran:

- Tipo de controlador gráfico.
- Tipo de fuente para un entorno MS-DOS basado en gráficos BGI.
- Identificación de cláusulas no deterministas.
- Especificación del tamaño del array interno.
- Despliegue de mensajes de error durante la compilación.
- Evitar la generación de código examinador.
- Alertas de errores de sintaxis.
- Habilitación de programación modular.

29

Sección de constantes

El papel de las constantes dentro del lenguaje Prolog es similar al que ejercen en otros lenguajes de programación, debido a que se emplean para representar cantidades, símbolos, entre otros.

Esta sección se encabeza con la palabra reservada Constants y las variables de este tipo se declaran <Nombre> = <Valor Asignado>.

Para mostrar un ejemplo de la declaración de esta sección se tiene el caso de una cantidad de 10 autos, lo cual siguiendo la sintaxis de Prolog, queda de la forma:

```
CONSTANTS
Autos = 10
```

Sin embargo, el uso de constantes presenta un conjunto de limitaciones:

- No existe distinción entre mayúsculas y minúsculas.
- No es posible realizar definiciones recursivas entre constantes.
- Toda constante debe ser definida antes de ser utilizada.
- Sólo se pueden declarar una vez.

30

Sección de dominios

La palabra dominio dentro del entorno de Prolog se refiere a los tipos de datos de las variables que serán utilizadas en el programa, específicamente en los predicados.

Prolog proporciona tipos de dato por defecto, no obstante, el programador puede personalizar dominios para hacer su programa más comprensible, es decir, puede crear nuevos tipos de variables basados en los existentes en Prolog de manera que los predicados se tornen más específicos en cuanto a la información que están manejando.

31

Sección de la base de datos

Como se ha mencionado en puntos anteriores, los programas pueden variar en su grado de complejidad, incluso llegando al punto en el que es imprescindible realizar modificaciones o actualizaciones en las bases de hechos creadas inicialmente, durante el tiempo de ejecución del sistema; para esto se encuentra a disposición la sección que emplea la palabra reservada Facts o Database.

32

Sección de los predicados

La sección de los predicados debe ir precedida por la palabra Predicates. Está dedicada a establecer el tipo de argumentos que utilizarán las cláusulas, por ende, es imprescindible que tanto los predicados como las cláusulas se encuentren estrechamente relacionadas, de lo contrario el sistema no podrá discernir ni tratar la información que se le está suministrando. En pocas palabras, en esta sección se definen las relaciones entre los objetos.

33

Sección de cláusulas

Esta sección es el centro del programa en Prolog, pues es donde se encuentran las bases de datos y las reglas que serán analizadas para satisfacer las metas establecidas. Se encabeza con la palabra Clauses.

Una regla tiene la siguiente sintaxis:

Head:- <sub goal>, <sub goal>.

34

Sección de meta u objetivo

Finalmente se tiene la sección de metas que Prolog debe satisfacer. Una meta u objetivo comparte una estructura similar a la de una regla, excepto por la ausencia del símbolo ":-". Una meta puede estar compuesta de varias submetas.

Ejemplo: Uso de las secciones básicas de un programa en Prolog

Para efectos didácticos se procede a desarrollar un ejemplo básico en el que se haga uso de las principales secciones de un programa escrito en Prolog.

35

AgentBuilder

AgentBuilder consiste en un conjunto de herramientas integradas en un software que permite a los desarrolladores un avance más rápido y fácil en los proyectos de creación de software inteligentes debido a que reduce el tiempo y el costo a la vez que incrementa el rendimiento de sistemas inteligentes robustos (Acronymics Inc., 2004).

Entre sus características más resaltables se encuentran:

- Fácil creación de agentes inteligentes.
- No requiere conocimientos especiales en diseño de sistemas inteligentes ni conexión a la red.
- Utiliza un lenguaje de alto nivel orientado a la programación de agentes.
- Permite la construcción de agentes con capacidades de operación autónomas y para la comunicación con otros agentes y usuarios.
- Entorno de desarrollo gráfico.
- Está basado en Java, lo que lo convierte en un generador de aplicaciones multiplataforma.
- Facilidad de integración con librerías en Java, C y C++.

36

Clips

Java Expert System Shell (JESS) es una variante de CLIPS escrito puramente en Java. JESS tiene dos usos fundamentales de los que destaca su papel como máquina de reglas. Un programa basado en reglas llega a tener hasta cien o incluso miles de reglas que JESS puede aplicar continuamente a los datos almacenados en forma de base de conocimiento.

Dos características fundamentales de JESS:

- Usa el método de encadenamiento hacia adelante para la inferencia del conocimiento.
- El modo de ingreso de comandos sólo es para el desarrollador no para el usuario final.

37

Jess

Java Expert System Shell (JESS) es una variante de CLIPS escrito puramente en Java. JESS tiene dos usos fundamentales de los que destaca su papel como máquina de reglas. Un programa basado en reglas llega a tener hasta cien o incluso miles de reglas que JESS puede aplicar continuamente a los datos almacenados en forma de base de conocimiento. El algoritmo que usa es una versión mejorada de Rete para emparejar las reglas con la base de conocimiento.

Dos características fundamentales de JESS:

- Usa el método de encadenamiento hacia adelante para la inferencia del conocimiento.
- El modo de ingreso de comandos sólo es para el desarrollador y no para el usuario final.

38

CAPÍTULO III: INTRODUCCIÓN A LA METODOLOGÍA COMMONKADS

39

INTRODUCCIÓN A LA METODOLOGÍA COMMONKADS

El término metodología hace referencia un conjunto de pasos o fases que se siguen con el fin de alcanzar un propósito; además que ofrece una estructura y mayor control, lo que facilita el desarrollo de un proyecto y su gestión.

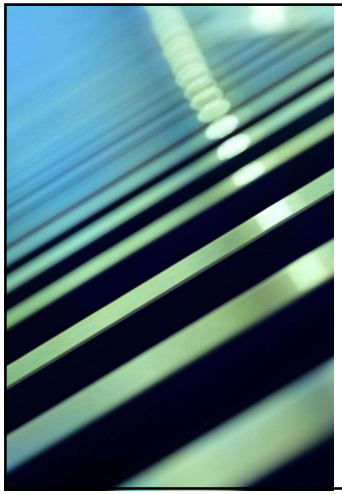
40

Aspectos generales de la Metodología CommonKADS.

Originalmente llamada Knowledge Acquisition Design System (KADS) para luego de su éxito, ser ampliando y adoptar el nombre CommonKADS, es una metodología enfocada en la construcción de SSBBCC, como lo son MIKE y Protégé.

Es el resultado de varios proyectos dentro del programa ESPRIT, que buscaba la innovación y aplicación de la informática avanzada en la Unión Europea, de manera que se obtuviera una calidad industrial en los SSBBCC de forma estructurada, manejable y repetitiva. Esta metodología se desarrolló en la Universidad de Ámsterdam junto a socios europeos (otras universidades, organizaciones de investigación y consultoría).

41



Principios fundamentales de la Metodología CommonKADS.

- La ingeniería del conocimiento consiste en construir diferentes modelos para la representación de ciertos aspectos del conocimiento humano.
- El principio del nivel de conocimiento: lo primero es concentrarse en los aspectos conceptuales del conocimiento para entonces proceder con la programación.
- El conocimiento posee una estructura interna estable en la que se distinguen tipos de conocimiento específicos y sus roles.
- Un proyecto de conocimiento debe ser gestionado basado en el aprendizaje de las experiencias propias en un controlado estilo de espiral.

42

Ciclo de Vida de la metodología CommonKADS.

El principio 4, tratado en la sección anterior, hace mención de que los proyectos en lo que se manipula conocimiento, deben seguir un estilo de espiral para su desarrollo, debido a la complejidad del conocimiento. No obstante, se siguen protocolos básicos de cualquier metodología:

- División del proceso en fases.
- Cada fase consta de actividades diferentes.
- Al final de cada fase se debe generar un producto.

43

Se reconocen las siguientes etapas en el proceso de producción de un SBC:

- Análisis: Su propósito es comprender el problema desde la perspectiva en la que se desea desarrollar. Productos característicos de esta fase: documento del proyecto en el que se incluyan: requerimientos, modelo conceptual, viabilidad y documentos de apoyo.
- Diseño: Se elabora la descripción funcional (comportamiento) del sistema, así como una descripción física en la que se hace énfasis en sus componentes.
- Implantación: Involucra la inserción de la aplicación dentro del contexto u organización para el que fue diseñado.
- Instalación: El sistema inicia operaciones dentro del entorno en que fue implantado.
- Uso: Se somete el sistema a ser manejado por los usuarios finales para la obtención de resultados.
- Mantenimiento: Corresponde a la etapa de mejoras o actualización del conocimiento.

44

Conjunto de modelos de la metodología CommonKADS.

CommonKADS cuenta con una serie de formularios enfocados en rasgos específicos para el modelado del conocimiento. Estos formularios son agrupados en tres categorías de acuerdo a su contribución para responder las tres preguntas fundamentales en el proceso de modelado del conocimiento: "¿Por qué?", "¿Qué?" y "¿Cómo?"

Nivel Contextual – Modelos de Contexto.

El nivel contextual se enfoca en responder la pregunta "¿Por qué?". De esta forma se debe aclarar el motivo por el cuál es necesario implementar un SBC para la solución de un problema.

Nivel Conceptual – Modelos Conceptuales.

La siguiente pregunta que debe responderse es "¿Qué?". El nivel conceptual va dirigido en determinar la estructura y naturaleza del conocimiento y la manera en que este es transmitido o intercambiado en el mundo real.

Nivel Computacional – Modelo de Diseño.

La última pregunta que se debe responder es "¿Cómo?", esto apoya al segundo principio que establece que es necesario definir primero un modelo conceptual para entonces proceder a la fase de programación del sistema.

45

CAPÍTULO IV: MODELADO DEL NIVEL CONTEXTUAL EN COMMONKADS

46

MODELADO DEL NIVEL CONTEXTUAL EN COMMONKADS

La primera etapa para el desarrollo de un SBC, siguiendo la metodología CommonKADS, abarca una serie de formularios enfocados en descubrir y formalizar aspectos relacionados al entorno en el que se planea insertar el SBC. Algunos de estos aspectos incluyen la identificación de problemas, oportunidades, soluciones y la factibilidad de estas.

El modelado a nivel contextual se encuentra subdividido en tres modelos: organizacional, tareas y agentes.

47

Modelo de la Organización de la metodología CommonKADS

Dentro del modelo organizacional se describe la estructura como tal del entorno especificando las funcionalidades de cada uno de sus elementos; proceso por el cual es probable detectar deficiencias como oportunidades para mejorar las distintas tareas con la implementación de un SBC.

Descripción de los problemas y posibilidades de mejora.

El primer formulario, OM-1, persigue como fin principal el contexto de la organización, sus problemas y sus probables soluciones; estos últimos requieren una perspectiva lo más realista posible, así como un entendimiento concreto y explícito desde una vista de negocios.

Descripción de los aspectos de interés de la organización.

El segundo estudio por realizar va de la mano con los elementos que poseen probabilidades de cambiar una vez se haya integrado el sistema en la organización, por ejemplo, el orden de los procesos, recursos, personal involucrado, entre otros

Descripción detallada de los procesos de interés.

En esta fase, se le da un vistazo más profundo al proceso que se esté estudiando. Un proceso es divisible en tareas más pequeñas y sencillas. Este proceso de análisis generalmente recurre al uso de diagramas de actividades basados en Unified Model Language (UML), de igual forma este diagrama funciona como guía en la sección de procesos del formulario OM-2.

Análisis de viabilidad.

El último paso dentro del modelo organizacional es el análisis de factibilidad, que no es más que un enfoque en los elementos principales de los cuatro formularios anteriores en los que se presta mayor atención en:

- Áreas de aplicación más prometedoras y las mejores soluciones.
- Beneficios versus costos.
- Disponibilidad de la tecnología requerida.
- Acciones que pueden impulsar el proyecto.

48

Modelo de Tareas de la metodología CommonKADS.

El estudio siguiente al modelo organizacional, es el modelo de tareas, este tiene un enfoque en las características de las tareas relevantes dentro del sistema. Es una forma de refinar los resultados obtenidos en los formularios OM. Sin embargo, antes de proceder a realizar el análisis correspondiente, es importante contar con la certeza de que se han identificado correctamente las tareas dentro del proceso del negocio. Para esto, se tienen a continuación las características de una tarea:

- Es una actividad orientada a una meta que añade valor a la organización.
- Maneja entradas y genera las salidas deseadas de forma estructurada y controlada.
- Utiliza recursos.
- Requiere y provee conocimiento y otras competencias.
- Ejecutada bajo estándares definidos y criterios de rendimiento.
- La ejecuta un agente (humano o software).

49

Descripción detallada de tareas.

- Los formularios anteriores proveen una visión del conocimiento y las tareas orientadas al entorno del negocio, no obstante, en el modelo de tareas ya es más factible añadir información como dependencias en el flujo de la información, los objetos manipulados, el control del tiempo; a su vez que se aplica un análisis estructurado propio de la ingeniería de software y una metodología orientada a objetos

Especificación del conocimiento.

- La especificación del conocimiento es una ampliación del formulario OM-4, además de que se concentra en los cuellos de botella y las posibles mejoras que requiera el sistema con mayor detalle.

50

Modelo de los Agentes de la metodología CommonKADS.

Dentro de los dos modelos anteriores se han estudiado las tareas desde la perspectiva de negocio y en la manera en que estas trabajan; por lo que en este último modelo se hace un paréntesis en los agentes encargados de realizar las tareas que ya fueron identificadas.

Descripción de los agentes.

El propósito del modelo de agentes es comprender los roles, capacidades y habilidades con las que deben contar los actores dentro de la organización con el fin de lograr sacar provecho al conocimiento que se busca sistematizar.

Documento de toma de decisiones sobre impactos y mejoras

El documento final de la etapa de modelado contextual en la metodología CommonKADS es el Organization, Task, Agent Models (OTA).

51

CAPÍTULO V: MODELADO DEL NIVEL CONCEPTUAL EN COMMONKADS

52

MODELADO DEL NIVEL CONCEPTUAL EN COMMONKADS

El análisis realizado mediante los formularios del modelo organizacional propicia una perspectiva más detallada del contexto en el que se planea implementar el sistema.

Estudiando los objetos de conocimiento, las tareas y los actores o agentes que las ejecutan. La siguiente etapa busca definir la estructura de la información que será empleada por el sistema, sin exponer la manera en que esta será introducida. En pocas palabras se especifica lo que hará el sistema, pero no la forma de hacerlo.

53

Modelo de Conocimiento de la metodología CommonKADS.

Se inicia el proceso de modelado del conocimiento, en el que se detallan aspectos como los tipos de conocimiento, su dominio y el nuevo conocimiento que se generará a partir del existente, todo esto sin caer en el detalle de implementación.

- Construcción del modelo de conocimiento de CommonKADS: Para la elaboración o construcción del modelo del conocimiento, se cuentan con tres fases que se describen a continuación:
- Identificación del conocimiento: Algo fundamental es verificar las fuentes de procedencia del conocimiento que será insertado en el sistema, en lo cual es recomendable la creación de un glosario en el que se albergue la terminología propia del conocimiento de manera que tanto los ingenieros, expertos y usuarios posean un lenguaje común.
- Especificación del conocimiento: La especificación del conocimiento se refiere al proceso de clasificación del conocimiento, definiendo las entradas y salidas, y a su vez el conocimiento inferido
- Refinamiento del conocimiento: Con las relaciones y características del conocimiento definidas, se procede a realizar simulaciones, de las cuales, si se obtienen resultados positivos, es decir, la estructura modelada es una aproximación lo bastante buena al sistema deseado, se da por finalizada la base del conocimiento.
- Documentación sobre el modelo de conocimiento: Terminadas las tres fases: identificación, especificación y refinamiento del conocimiento, sólo queda reunirlo todo en un solo formulario, el KM-1

54

Categorías de conocimiento.

En el formulario KM-1 se hace mención de los tipos de conocimientos sobre el dominio (base), inferencia y tareas. Cada uno de estos conocimientos cuenta con sus características particulares que se detallan en los siguientes puntos:

Conocimiento del dominio.

- El conocimiento del dominio hace referencia a toda la información que será introducida en el sistema perteneciente a un área específica de estudio, por ejemplo: diagnósticos de enfermedades, en donde se usará una terminología propia de los especialistas en el área entre los que se incluyen nombres de enfermedades, síntomas, entre otros

Conocimiento sobre inferencias.

- Esta categoría del conocimiento recoge lo que vienen siendo los bloques de instrucciones que la máquina utilizará para manipular el conocimiento del dominio, de manera que se genere nuevo conocimiento a partir del existente

Conocimiento sobre tareas.

- El conocimiento sobre las tareas incluye la descripción de las metas u objetivos que debe cumplir dicha tarea y la estrategia para lograrlo.

55

Tipos de tareas proporcionadas por la librería de CommonKADS.

La librería CommonKADS proporciona una clasificación para las tareas que realiza un SBC. Se agrupan en dos tipos: analíticas y sintéticas. La diferencia central entre los dos tipos yace en el tipo de sistema en el que son llevadas a cabo.

Tipos de tareas analíticas.

- Se reconocen como tareas del tipo analítico aquellas que no requieren de que el sistema se encuentre construido como tal, sino que solo basta con ingresar cierta información sobre el sistema para obtener alguna caracterización del sistema en base a esa información.

Tipos de tareas sintéticas.

- Para el caso de las tareas sintéticas, es imprescindible que el sistema propuesto ya se haya construido, pues las entradas a estas tareas dependen de lo que el sistema les pueda suministrar.

56

Modelo de Comunicación de la metodología CommonKADS.

El modelo de comunicación es en el que se describen las transacciones o intercambios de datos e información entre las tares realizadas por distintos agentes.Este proceso de modelado conlleva tres fases al igual que el modelo del conocimiento, y se detallan a continuación:

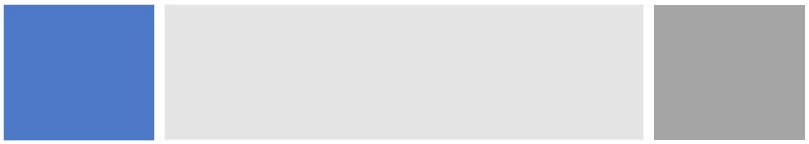
Plan de comunicaciones. Es la parte más sencilla del modelo, en este se explican o describen de forma general los agentes que requieren comunicarse durante una serie de tareas dadas y las necesidades que se deben cubrir durante ese intercambio de información.

Transacciones. El término transacción dentro del contexto de la metodología CommonKADS, se refiere al intercambio de información entre agentes durante la realización de una tarea dada

Especificaciones del intercambio de información. Se procede a realizar un refinamiento de la información recabada de la transacción por el formulario CM-1. En la tabla 13, se presenta el formulario CM-2, en el que se añaden elementos de control, sintaxis y otras explicaciones que guardan relación con la estructura interna de la transacción para llegar a las conclusiones que se han descrito en el CM-1.

57

CAPÍTULO VI: MODELADO DEL NIVEL COMPUTACIONAL EN COMMONKADS



58

MODELADO DEL NIVEL COMPUTACIONAL EN COMMONKADS

Luego de efectuado los modelos de conocimiento y comunicación, se pasa a la etapa que corresponde al diseño del sistema, o bien, modelado a nivel computacional.

Durante este proceso se pretende mantener la información recolectada sobre el conocimiento a medida que se diseña la arquitectura del sistema, esto implica que el modelo del conocimiento contiene los requerimientos funcionales con los que debe contar el sistema; por otra parte, el modelo de comunicación especifica los requerimientos no funcionales, es decir, todo lo que involucra la interacción con factores externos al sistema.

A lo largo del capítulo se describe en qué consiste la estructura del modelo del diseño que sigue la metodología CommonKADS y su desarrollo paso a paso.

59

Modelo de Diseño de la Metodología CommonKADS

Lo primero a destacar del modelo de diseño en CommonKADS es el principio de structure-preserving design que lo rige. Este principio hace énfasis en no hacer cambios o modificaciones en la información recolectada en los documentos previos a esta fase, pues de lo contrario no se logra mantener la transparencia y se pierde la posibilidad de aplicar métodos de mantenimiento eficientes para el sistema; por otra parte, seguir este principio propicia una alta calidad en el diseño. En el siguiente punto se profundiza un poco sobre el principio de structure-preserving design.

Principio Structure-Preserving Design en CommonKADS.

La idea central de este principio de preservar la estructura del diseño se enfoca en mantener el contenido y estructura de la información del dominio que ha sido presentada en los informes anteriores de forma que este conocimiento pueda ser extraído o recuperado del sistema final, así como las relaciones entre objetos de conocimiento existentes.

60

Proceso de Diseño en CommonKADS.

Todo proceso de diseño inicia con la especificación de la arquitectura del software de manera general, una vez esto se haya definido se inicia con una especificación más detallada de la arquitectura. En CommonKADS este proceso consta de cuatro fases que se describen a continuación:

Diseñar la arquitectura del sistema.

Dentro de este primer paso hay que tomar en consideración tres lineamientos básicos de cualquier arquitectura: El sistema se descompone en subsistemas, El control general, La descomposición de los subsistemas en módulos del programa.

Identificar la plataforma de implantación.

La plataforma de implantación se refiere al software sobre el que se construirá el sistema y el hardware que lo soportará, es preciso seleccionar cuidadosamente ambos, pues en caso contrario habrá serias afectaciones en los pasos posteriores del diseño.

Especificar los componentes de la arquitectura.

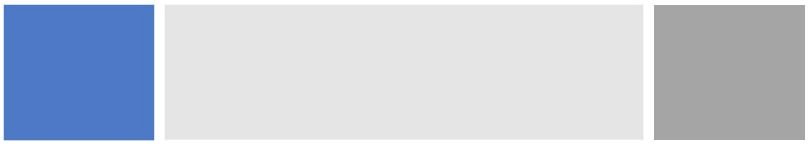
Este paso se dedica a ampliar detalles sobre los componentes de la arquitectura, es decir, brindar más información sobre las interfaces que deben existir entre los subsistemas o módulos, sobre todo los aspectos relacionados a las opciones de toma de decisiones creadas por el diseñador y que son aplicables dentro del sistema.

Especificar la aplicación dentro de la arquitectura.

Finalmente, se toman los elementos: tareas, bases de conocimiento, transacciones, entre otros; para ser definidos y ubicados dentro de la arquitectura que se ha diseñado.

61

CAPÍTULO VII: REPRESENTACIÓN DEL CONOCIMIENTO MEDIANTE LÓGICA DE PREDICADOS.



62

REPRESENTACIÓN DEL CONOCIMIENTO MEDIANTE LÓGICA DE PREDICADOS.

El elemento principal para la construcción de un SBC es el conocimiento, sin embargo, no es posible almacenar este conocimiento en lenguaje común para cualquier persona, debido a que una computadora o sistema informático no se encuentra en la capacidad de procesar este tipo de información, por ende, es necesario recurrir a técnicas o métodos que permiten generar una representación del conocimiento legible y manipulable por las máquinas en las que se pretende implementar el sistema.

En el capítulo se exponen las representaciones más comunes: lógica de predicado y lógica proposicional.

63

Representación del Conocimiento.

La representación del conocimiento consiste en escribir en un lenguaje que las computadoras puedan procesar, sentencias que describan el conocimiento que se tiene sobre un dominio específico. Los lenguajes empleados para esta tarea cuentan con su propia sintaxis, estructura y reglas.

Esta es la etapa que representa el mayor reto al momento de desarrollar un sistema dentro del ámbito de la inteligencia artificial como lo es un SBC.

64

Lógica simbólica o formal.

La lógica simbólica es una ramificación de la lógica que se enfoca en la creación de lenguajes formales, sistemas semánticos y deductivos que ofrezcan la alternativa de efectuar sobre ellos análisis del tipo matemático, de los que generalmente se obtienen teoremas.

Para comprender mejor la lógica simbólica formal, es necesario tener presente el concepto de lenguaje formal. Un lenguaje formal es aquel que cuenta con símbolos, reglas de escritura específicos o sintaxis y una semántica.

65

Representación del conocimiento mediante lógica de predicados.

Mediante la lógica de predicados o lógica de primer orden se pueden representar expresiones más complejas del lenguaje natural, permitiendo así su uso en el proceso de inferencias, sobre todo aquellas que varían de acuerdo al valor que se le otorgue a sus elementos. La lógica de predicado cuenta con una serie de elementos que se listan a continuación:

- Variables proposicionales: Aquellas variables que solo pueden tomar un valor de verdadero o falso. Se denotan con letras minúsculas.
- Constantes: Se usan para referirse a seres del mundo real o ideal. De modo convencional, utilizan las primeras letras del abecedario.
- Términos: Corresponden a los sustantivos de la oración y pronombres de la oración. Este concepto incluye a los dos anteriores: variables y constantes.
- Cuantificadores: Encargado de definir para cuantos con la que un predicado es verdadero. Corresponden a las palabras “todo”, “algunos”, “nunca” y demás expresiones similares.
- Predicados: Siguiendo el concepto conocido en español, corresponde a las afirmaciones sobre propiedades o relaciones existentes de los términos.

66

Inferencia y razonamiento.

La inferencia y razonamiento parten de un elemento principal para todo SBC, la base de conocimiento. Esta base de conocimiento se encuentra conformada por hechos y reglas o sentencias escritas en lenguaje formal, comprensible a nivel computacional.

67

Métodos básicos de razonamiento.

Para obtener el nuevo conocimiento existen dos métodos básicos del razonamiento:

Razonamiento hacia adelante: Partiendo de hechos, se le proporcionan una serie de parámetros, que van siendo evaluados para llegar a una conclusión o nuevo hecho.

68


Reglas para transformar expresiones en cláusulas lógicas.
Una cláusula lógica está conformada por un conjunto finito de literales, donde los literales corresponde a fórmulas atómicas o declaraciones combinadas por conectivas lógicas, como las empleadas en la lógica proposicional.