ISMM

"Dr. Antonio Núñez Jiménez"
Facultad Metalurgia-Electromecánica
Carrera Ing. Informática

Trabajo de Diploma para optar por el Título de Ingeniero en Informática

Métodos de Adaptación en el Razonamiento Basado en Casos.

Autora:

Yisel Rivera Samón.

Tutora:

Dra. Yiezenia Rosario Ferrer.

Moa, Holguín, 2009
"Año del 51 Aniversario de la Revolución"

DECLARACIÓN DE AUTORÍA

Yo, Yisel Rivera Samón, estudiante del Instituto Superior Minero Metalúrgico (ISMM), Dr. Antonio Núñez Jiménez, declaro que soy la única autora de la presente investigación titulada: "Métodos de Adaptación en el Razonamiento Basado en Casos" y autorizo a ser uso de la misma en su beneficio al ISMM.

Para que así conste firmamos la presente a los _____ días del mes de _____ del 2009.

__Yisel Rivera Samón_____
Nombre completo del primer autor

__Yiesenia Rosario Ferrer_____
Nombre completo del primer tutor

Pensamiento

"…la lógica formal es algo más que una tecnología: es la moral del pensamiento y del discurso. Enseña el ejercicio de la honestidad total, tanto en la expresión como en la justificación. "

(I. M Bochenski)

Dedicatoria

A mi mamá, porque no existe en el mundo palabra capaz de definir cuanto significa, cuanto debo agradecer por ser hoy quien soy, por el amor que me ha brindado, porque a usted me debo y dedico este trabajo.

A mi familia por toda la dedicación que me brindó, la fe y la confianza.

A mí querida Amparo por todo su apoyo incondicional, por guiarme en el camino de la razón y por todas las cosas lindas que hizo por mí.

Agradecimientos

A mi tutora, por guiarme en todo momento y por toda su ayuda.

A mi mamá por proveerme siempre lo necesario para cumplir mis metas, por mantenerme firme y siempre darme apoyo.

A mí estimada Amparito porque tengo que darle gracias a Dios por haberla encontrado en mi camino, por alentarme, aconsejarme, apoyarme, cuidarme y darme fuerzas para continuar.

A mi querido hermano Luis Henri quien siempre ha sabido conservar este amor de hermandad que nos une.

A mi querido papá Jorge Luis por toda la dedicación que me ha dado.

A mi familia, por el apoyo que de ellos he recibido.

A mi querida abuelita Elidia, mi tía Deisy y mi tío Israel, por la gran dedicación que siempre me brindaron y por toda la fe que me dieron.

A mis amigos, quienes han sabido mantenerse cerca en cada paso de la vida.

A quienes días tras días forjamos este grupo de amistad, a Virgen, Yudith, Alina, Amparo.

A Yomaida, Magday, Yoannis, Maida por apoyarme, ayudarme y darme fe en mi trabajo.

A Virgen, Ariel, Yeidel, Jhonlier por tenderme la mano en el momento que más la necesitaba.

A todos aquellos que de una forma u otra han contribuido con mi formación y la realización de este trabajo.

Muchas Gracias

RESUMEN

Razonamiento Basado en Casos significa resolver un nuevo problema recordando una situación similar previa y reutilizando su información y conocimiento.

El Razonamiento Basado en Casos es una manera de razonar haciendo analogías. Se ha argumentado que el Razonamiento Basado en Casos es usado por las personas para solucionar problemas cotidianos, además de ser más que un método poderoso para el razonamiento de computadoras. Más radicalmente se ha sostenido que todo Razonamiento es Basado en Casos por que está basado en la experiencia previa.

Los sistemas con Razonamiento Basado en Casos realizan adaptación para llegar a la solución de un determinado problema. La adaptación que realizan para la solución de un problema es a partir de las soluciones de problemas similares pasados. Este proceso requiere que el caso recuperado sea modificado y así darle una correcta solución al nuevo problema. Existen diferentes métodos de adaptación y ante el planteamiento de un nuevo problema se debe escoger el método de adaptación más adecuado para dar solución al mismo.

El presente trabajo de diploma propone un estudio de algoritmos de adaptación en el Razonamiento Basado Casos, analizar cómo realizan la adaptación, implementar los métodos escogidos y realizar comparaciones.

ABSTRACT

Case Based Reasoning means to solve a new problem by recalling a similar situation before and reusing information and knowledge.

The Case Based Reasoning is a way of reasoning by analogy. It has been argued that the Case Based Reasoning is used by people to solve everyday problems, as well as more than a powerful method for computer reasoning. More radically it has been argued that everything is case-based reasoning that is based on previous experience.

Systems with Case-Based Reasoning to perform adaptation to solve a particular problem. Adaptation made to solve a problem from the solutions of similar past problems. This process requires that the case is retrieved and modified to give a correct solution to the problem again. There are different ways to adapt and to approach a new problem is to choose the most suitable adaptation technique to solve it.

This work proposes a study of degree of adaptation algorithms in the Case Based Reasoning, analyze and perform the adjustment, the means chosen to implement and make comparisons.

ÍNDICE DE MATERIAS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Desde el surgimiento mismo de la computación y a lo largo de toda su evolución se ha intentado modelar o simular el pensamiento humano y los procesos que ocurren en él. En los inicios solo se trataba de representar en las computadoras el pensamiento estructurado, los algoritmos de cálculos que podían definirse claramente como un conjunto de pasos que podían ser interpretados por las máquinas y de cierta forma sustituir o contribuir a elevar la eficiencia del ser humano en este tipo de actividades.

La introducción de la informática en todos los campos del saber humano adquiere cada día una mayor relevancia. Un área particularmente prometedora de la computación lo constituye la **Inteligencia Artificial** que ha desarrollado técnicas de aplicación en ramas específicas que la han transformado de ciencia puramente académica en ciencia experimental.

La Inteligencia Artificial se dedica a la creación de hardware y software que imita el pensamiento humano. Su principal objetivo es llevar a la computadora las amplias capacidades del pensamiento humano y, para ello, se convierten en “entes inteligentes” con la creación de software que les permite imitar algunas de las funciones del cerebro humano en aplicaciones particulares. El fin no es reemplazar al hombre, sino proveerlo de una herramienta poderosa para asistirlo en su trabajo.

Dentro de las metodologías de Inteligencia Artificial, el **Razonamiento Basado en Casos** ocupa un importante lugar ya que facilita el uso de la experiencia acumulada para la toma de decisiones sobre las nuevas situaciones que se presenten.

El Razonamiento Basado en Casos es un paradigma de resolución de problemas, pero son precisamente las diferencias con el resto de los acercamientos de la Inteligencia Artificial las que lo hacen tan especial. En lugar de confiar únicamente en el conocimiento general del dominio del problema, o realizar asociaciones a lo largo de relaciones entre descripciones del

problema y conclusiones, este paradigma es capaz de utilizar conocimiento específico de experiencias previas, es decir, situaciones de un problema concreto (casos).

Ante el planteamiento de un problema no abordado con anterioridad, se intenta localizar un caso pasado similar y adaptar su solución a la situación del problema nuevo. De esta adaptación podemos obtener una nueva experiencia a la hora de resolver problemas con ciertas similitudes, lo que nos lleva a una segunda diferenciación del Razonamiento Basado en Casos con respecto al resto de tendencias, el aprendizaje incremental, ya que las nuevas adaptaciones se almacenan como nuevos casos, relacionados, y disponibles para comparaciones futuras.

El modelo de razonamiento que sugiere el Razonamiento Basado en Casos incorpora una forma de adquirir conocimiento, un método de resolución de problemas que permite resolver nuevos problemas a partir de otros precedentes y un enfoque de aprendizaje pues el nuevo caso resuelto es guardado en la Base de Casos para futura reutilización.

Existe un gran número de problemas donde el Razonamiento Basado en Casos puede ser el único método de solución o en ocasiones el mejor. En diseño, por ejemplo, el problema a resolver queda definido por un conjunto de restricciones para las cuales muchas veces no existe una solución correcta que satisfaga todos los requerimientos. En estos casos no existe un algoritmo que permita hallar la solución. Otras veces sucede que el espacio de soluciones es muy grande, por lo que una búsqueda en el mismo podría consumir gran cantidad de tiempo. Ese mismo problema, en ocasiones, es muy grande y difícil de descomponer (las partes pueden estar muy ligadas entre sí) en partes que se puedan solucionar de manera independiente para luego unir las soluciones parciales en una final.

La adaptación de un caso puede ser llevada a cabo ya sea por la transformación de un caso solo para ajustarlo a los requerimientos de la nueva situación, o mediante la composición apropiada de partes de varios casos. La adaptación a menudo es muy difícil y puede ser eludida del todo, es por esto que ella ha recibido mucha menos atención que la

recuperación. La evaluación y modificación de los métodos de adaptación son importantes para construir un sistema poderoso.

Muchos sistemas que utilizan Razonamiento Basado en Casos realizan la adaptación para llegar a la solución del problema planteado. Si no realiza una buena adaptación o si no se escoge el método de adaptación más adecuado, el nuevo caso resuelto puede tener fallos.

La adaptación de la solución de un problema a partir de las soluciones de problemas similares debe ser una manera eficiente y directa de encontrar la solución para el problema dado, pero en la práctica no es tan sencillo.

La adaptación sigue siendo el paso más complejo del Razonamiento Basado en Casos, pues incorpora inteligencia que, de no ser así, sería un mero proceso de reconocimiento de patrones.

Otro aspecto fundamental es que los sistemas que emplean Razonamiento Basado en Casos, para realizar la adaptación manipulan un solo método de adaptación, no un conjunto de ellos, es decir, cada sistema implementa un métodos de adaptación dependiendo del dominio en el que se está trabajando, pero tal vez exista otro método que resuelva el problema de una manera más eficiente.

El presente trabajo de diploma pretende dar solución a la situación problemática expuesta, por lo que se plantea como **problema:** un proceso de adaptación deficiente en el Razonamiento Basado en Casos provoca que no se obtengan las soluciones adecuadas.

Este problema se enmarca en el **objeto de estudio** al Razonamiento Basado en Casos. Este objeto nos delimita como **campo de acción:** los métodos de adaptación.

Para dar solución al problema se propone el siguiente **objetivo general**: realizar un análisis comparativo de los métodos utilizados para la adaptación en el Razonamiento Basado en Caso.

Como **objetivos específicos** se plantean los siguientes:

1. Presentar una visión general de los métodos utilizados para la adaptación en el Razonamiento Basado en Casos.
2. Implementar los algoritmos seleccionados.
3. Comparar los resultados**.**

Para el desarrollo de este proyecto se plantea la siguiente **hipótesis**: el estudio de los diferentes métodos de adaptación utilizados en el Razonamiento Basado en Casos, permitirá seleccionar métodos adecuados para la construcción de sistemas con Razonamiento Basado en Casos.

Para cumplir los objetivos y resolver la situación problemática presentada, se ejecutaron las siguientes tareas:

- Revisar bibliografía de la teoría y la tecnología del Razonamiento Basado en Casos.
- Análisis de los métodos de adaptación.
- Elaboración e implementación de los algoritmos de los métodos seleccionados.
- Diseño de los experimentos y análisis de los resultados.

El presenta trabajo está conformado por Resumen, Introducción, tres capítulos, Conclusiones, Recomendaciones, Bibliografía y Anexos:

Capítulo1. Fundamentación Teórica**.**
En este capítulo se expone la definición de Razonamiento Basado en Casos, así como una breve descripción de características y conceptos imprescindibles que le dan base a la investigación.

Capítulo2. Herramientas y Métodos.
En este capítulo se abordará la descripción de los algoritmos de cada método seleccionado que se utilizan en la adaptación, así como los detalles de la implementación, en que lenguaje es implementado y por que se escogió el mismo.

Capítulo3. Experimentos y Resultados.

En este capítulo se expone todo lo relacionado con los resultados obtenidos. Así como los criterios empleados para realizar la comparación de los algoritmos.

Fundamentos Teóricos

1.1 Introducción

El **Razonamiento Basado en Casos RBC** (en inglés Case-Based Reasoning, abreviado CBR) es una técnica de la Inteligencia Artificial (abreviado IA) que intenta llegar a la solución de nuevos problemas de forma similar como lo hacen los seres humanos utilizando la experiencia acumulada hasta el momento en acontecimientos similares [6].

De manera más abarcadora podemos decir que el Razonamiento Basado en Casos significa resolver problemas a partir de experiencias precedentes (casos), adaptando soluciones antiguas para resolver problemas nuevos, o recuperando casos anteriores para iluminar aspectos de la situación actual. Esto implica una concepción bastante alejada de la mayoría de los sistemas de IA construidos hasta ahora. Hace de su última etapa, el aprendizaje, el centro de todo el proceso analógico.

Cuando un individuo se enfrenta a un nuevo problema comienza por buscar en su memoria experiencias anteriores similares a la actual y a partir de ese momento establece semejanzas y diferencias y combina las soluciones dadas con anterioridad para obtener una nueva solución. Este proceso es intuitivo y la persona lo realiza prácticamente sin darse cuenta.

Una vez que la persona tiene situadas un grupo de situaciones anteriores similares a la actual, analiza las variantes que se presentan en la nueva situación y cómo puede dar respuesta a estos cambios.

El Razonamiento Basado en Casos es el proceso de solucionar nuevos problemas basándose en las soluciones de problemas anteriores.
Un nuevo problema se compara con los casos almacenados previamente en la Base de Casos y se recuperan uno o varios casos. Posteriormente se utiliza y evalúa una solución sugerida, por los casos que han sido seleccionados con anterioridad, para tratar de aplicarlos al problema actual.

El RBC resuelve nuevos problemas adaptando los casos relevantes de su biblioteca. Se preocupa por el estudio de los mecanismos mentales necesarios para repetir lo que se ha hecho o vivido con anterioridad, ya sea por uno mismo, o ya sea por casos concretos recopilados en la bibliografía o en la sabiduría popular. En concreto, un sistema de este tipo debe:

- Extraer de los datos de entrada las características que mejor puedan utilizarse.
- Recuperar los casos relevantes basándose en la semejanza con la entrada, seleccionar el mejor caso(s).
- Adaptar los caso(s) para que cumplan las especificaciones del problema.
- Evaluar la solución con vistas a determinar si debe ser reparada o si puede ser directamente incorporada a la memoria de casos.
- En el caso de que hubiera habido reparación, recordarla indexándola convenientemente; y, finalmente,
- Realizar las generalizaciones pertinentes en función de los nuevos casos disponibles.

1.2 Historia

En los últimos años, el RBC ha experimentado un rápido crecimiento desde su nacimiento en Estados Unidos. Lo que sólo parecía interesante para un área de investigación muy reducida, se ha convertido en una materia de amplio interés, multidisciplinar y de gran interés comercial.
A continuación se mencionan algunos sistemas que utilizan RBC en el ámbito internacional [18] y nacional.

1.2.1 Ámbito Internacional

- ✓ **Script** [29]**:** Propusieron almacenar la información de cada situación con scripts. Los Scripts fueron propuestos como modo de representación de la memoria conceptual, definiendo eventos tipo tales como ir a un restaurante o visitar al médico. No era una teoría completa.
- ✓ ReMaind, programa DARPA.
- ✓ Roger Schank[28] y su grupo en la Universidad de Yale crearon, a principios de los años 80:
  - ❖ El papel de la memoria en el razonamiento.
- ✓ Los primeros sistemas académicos [26]:
  - ❖ **Cyrus**[1] [15]**.**
  - ❖ **Mediator** [28].
  - ❖ **CHEF**: Planificador basado en casos cuyo propósito es la creación de recetas de comida [12].
  - ❖ **Persuder.**
  - ❖ **Casey**: Programa de diagnostico basado en casos. Este realiza un diagnóstico de enfermedades del corazón.
  - ❖ **Julia**: Es un sistema de diseño basado en casos que opera en el dominio de planificación de comidas.

- ✓ **JUDGE**: Modelo basado en casos de sentencias criminales. Razona sobre asesinatos, muertes accidentales y otros.
- ✓ **PLEXUS**: Adapta planes viejos para nuevas situaciones.
- ✓ Bruce Porter y su grupo en la Universidad de Texas en Austin:

Integración de casos y conocimiento general, el sistema **Protos:** Programa que implementa clasificación y adquisición de conocimiento basado en casos. Trabaja en el dominio de los desordenes auditivos.

- ✓ **COACH**: Genera una nueva jugada de fútbol mejorando otras anteriores.

[1] Janet Kolodner, "Reconstructive Memory: A Computer Model," *Cognitive Science* 7 (1983).

- ✓ **DECIDER**: Ayuda a los estudiantes a comprender o a resolver problemas pedagógicos seleccionando y presentando casos apropiados de una base de datos que responde a los objetivos del estudiante.
- ✓ Edwina Rissland y su grupo en la Universidad de Massachusetts en Amhearst aplicaciones al dominio de las decisiones legales, el sistema **Hypo:** Es un razonador interpretativo basado en casos que trabaja en el dominio de la ley.
- ✓ **BATLE**: Trabaja en el dominio de planificación de guerra en tierra.
- ✓ 1990, primera aplicación en la industria **CLAVIER^2:** Trabaja en el dominio de carga de un autoclave para la contracción de piezas [25].
- ✓ **CASCADE**: Fallos en el sistema operativo VMS.
- ✓ **BOLERO** [1]: Sistema que combina RBC y reglas.
- ✓ **TOTLEC**: Planificación de manufacturas complejas como la detección de errores durante la fase de diseño y advierte al usuario sobre ese diseño defectuoso.
- ✓ **PAKAR** [32]: Identifica causas en los defectos de construcción de edificios.
- ✓ **KICS**: Regulación en la construcción. Acumula casos históricos de interpretación de la regulación.

1.2.2 Estudios en el País

- Razonamiento Basado en Casos en Ciencias Médicas sobre plataforma Web [8].

- La aplicación denominada Shaftwizard, que ayuda a elaborar tecnologías de maquinado para piezas tipo eje, está introducida en las empresas INPUD, Combinado Textil, Minerva, y Aguilar Noriega, de Santa Clara [3][20].

- El sistema SISI se ha aplicado en el diagnóstico de infarto cardiaco y epilepsia, en el Centro de Cibernética Aplicada a la Medicina (CECAM), en el diagnóstico de inmunodeficiencias en la Facultad de Ciencias Médicas de Matanzas, y para el pronóstico de malformaciones cardiovasculares en recién nacidos, en el hospital materno Mariana Grajales de Villa Clara [3].

- Uso del Razonamiento Basado en Casos combinado con técnicas estadísticas para el diagnóstico de la Hipertensión Arterial [5].

- Sistema Experto (SE) para el diagnóstico y tratamiento del Embarazo Ectópico. Es un SE híbrido que combina el Razonamiento Basado en Casos y las reglas de producción [27].

- Modelo de un Sistema Basado en Casos para el diagnóstico del Síndrome de Maloclusión [19].

1.3 Ciclo del Razonamiento Basado en Casos

El proceso del RBC puede ser visto como un ciclo (figura 1.2). Este ciclo también es conocido como el ciclo de las cuatro REs (a partir de sus nombres en inglés).

1. Recuperar (en inglés: Retrieve) el caso o casos más similares.
2. Adaptar o Reutilizar (en inglés: Reuse) la información y el conocimiento de ese caso para resolver el problema.
3. Revisar (en inglés: Revise) la solución propuesta.
4. Guardar (en inglés: Retain) las partes de esta experiencia que se consideren útiles para resolver futuros problemas.

Un problema nuevo se resuelve recuperando uno o más casos previos, reutilizando el caso de una manera u otra, evaluando la solución propuesta, y aprendiendo la nueva experiencia por medio de su incorporación a la base de conocimiento existente (*Base de Casos*).

[2] Bill Mark, "Case-Based Reasoning for Autoclave Management," *Proceedings of the Case-Based Reasoning Workshop* (1989).

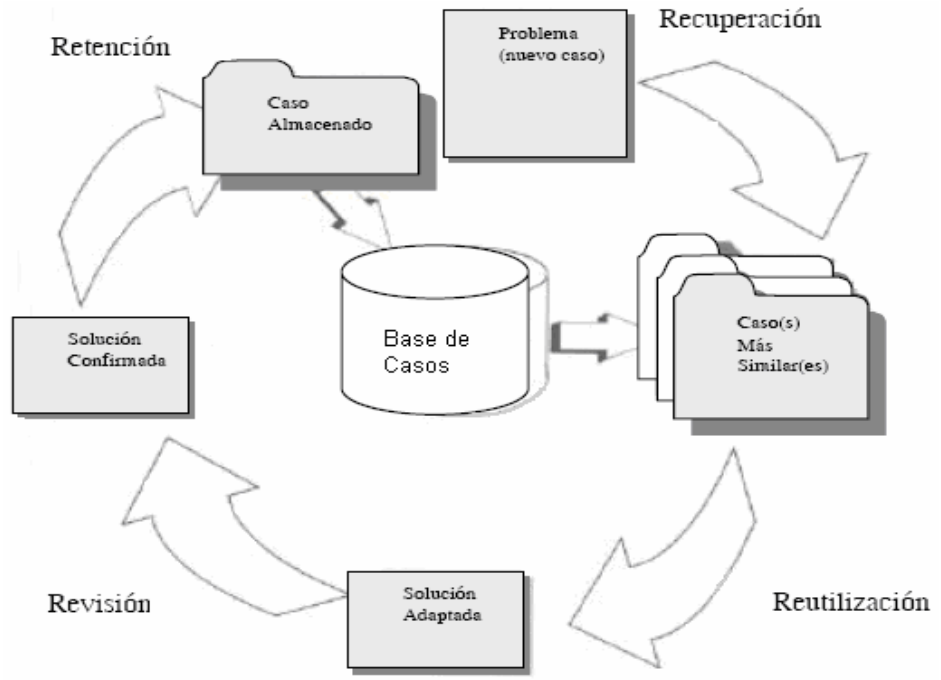

Fig 1.2 Diagrama conceptual del proceso de Razonamiento Basado en Casos.

La figura anterior se puede observar el proceso del RBC. Una descripción de un problema define un nuevo caso, que se usa en la Recuperación de un caso de entre la colección de casos pasados. El caso recuperado se combina con el nuevo caso para, a través de la Reutilización, proponer un caso que sea una solución al problema inicial. En una fase de Evaluación se verificará el éxito de la solución, por ejemplo siendo comprobada en el mundo real (habitualmente por un agente humano), y se reparará si falla. Durante el Almacenamiento, la experiencia útil se guarda para una futura reutilización, y el caso base se actualiza por un nuevo *caso aprendido*, o por una modificación de algunos casos existentes.

Los cuatro procesos del RBC no son tareas únicas, cada uno de ellos implica llevar a cabo una serie de tareas más específicas.

Esta investigación solo está centrada en el proceso de Adaptación o Reutilización, pues es la etapa encargada de dar solución al problema.

1.3.1 ¿Qué es un "CASO"?

Un caso se puede considerar un registro del problema. La información que se almacena sobre un problema depende tanto del dominio como del propósito donde el caso es usado.

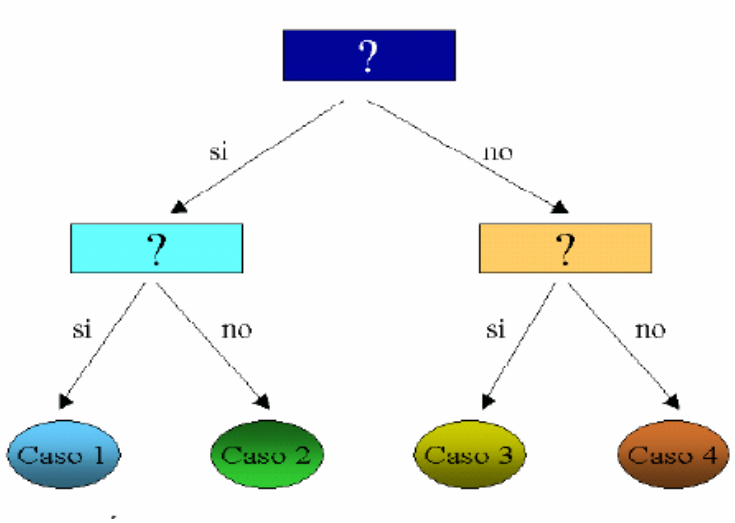

Fig 1.1 Árbol de decisiones para la clasificación de los casos.

El conocimiento que almacena un caso es específico, no abstracto y todo conocimiento relacionado se encuentra cerca de la Base de Casos, esto quiere decir que el conocimiento que necesitamos para resolver un problema específico lo encontraremos agrupado en unos pocos casos o incluso en uno.

En los sistemas con RBC la Base de Casos es la memoria de casos almacenados. Desde el punto de vista del informático, la Base de Casos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

1.4 Elementos básicos del Razonamiento Basado en Casos

Los elementos básicos de un sistema con RBC son (ver figura 1.3):

- **Representación del Conocimiento:** En un sistema con RBC, el conocimiento es representado principalmente en forma de casos que describen experiencias concretas.

- **Medida de similitud:** Encontrar un caso relevante para el problema actual.

- **Adaptación:** Situaciones pasadas representadas como casos difícilmente serán idénticas a las del problema actual. Sistemas con RBC tienen mecanismos y conocimiento para adaptar los casos recuperados completamente, para verificar si satisfacen las características de la situación presente.

- **Aprendizaje:** Para que un sistema se mantenga actualizado y se desarrolle continuamente, siempre que el resuelva un problema con éxito, deberá ser capaz de recordar esa situación en un futuro como una característica de un nuevo caso.

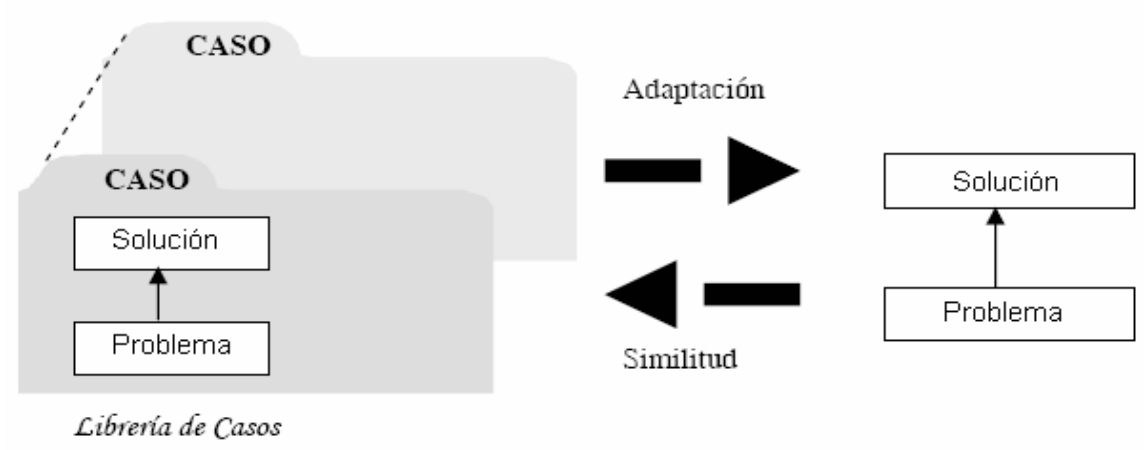

Fig1.3 Modelo básico del enfoque RBC

1.5 Ventajas, inconvenientes y problemas actuales de los RBC

Ventajas

1. El esfuerzo en la solución de problemas puede ser capturado para ahorrar trabajo en el futuro.
2. Experiencias previas que hayan sido exitosas pueden ser utilizadas para justificar nuevas soluciones.
3. Experiencias previas que no hayan sido exitosas se pueden utilizar para anticipar problemas.
4. La comunicación entre el sistema y los expertos se realiza en base a ejemplos concretos, es decir el sistema explica sus decisiones citando precedentes.
5. El RBC trabaja a partir de bases de datos existentes, no se requieren entrevistas con los expertos, simplificándose la adquisición del conocimiento.
6. El RBC es un algoritmo de aprendizaje incremental, el aprendizaje tiene lugar tan pronto como un nuevo ejemplo esta disponible, sin excesivo costo computacional.
7. El RBC permite proponer soluciones para los problemas rápidamente, evitando el tiempo necesario para derivar respuestas desde el estado inicial de un proceso de búsqueda de soluciones. Esta ventaja se manifiesta principalmente en situaciones donde un sistema basado en reglas, por ejemplo, hubiese requerido realizar una larga cadena de inferencias para alcanzar una solución.
8. El RBC permite proponer soluciones en dominios que no se comprenden completamente.
9. Los casos ayudan a focalizar el razonamiento sobre las partes importantes de un problema señalando que rasgos del problema son importantes.
10. El RBC es aplicable a un amplio rango de problemas.

Inconvenientes

1. Resulta difícil encontrar una estructura apropiada para describir el contenido de un caso y decidir cómo la memoria de casos debe ser organizada e indexada para un almacenamiento, recuperación y rehúso efectivos y eficientes.

2. Cuando el modelo del conocimiento general del dominio es incompleto, presenta información imprecisa, etc., resulta difícil la integración del mismo a la estructura de la base de casos.
3. Su principal limitación radica en la capacidad de representación de los valores de los rasgos, aspecto que dificulta su aplicación en situaciones que requieren representaciones de casos complejas como por ejemplo las que se presentan cuando existe un alto grado de interrelación entre los rasgos.

También hay que señalar que la definición de las funciones de semejanza y los mecanismos de selección de rasgos para cada problema específico son problemas abiertos en este campo, cuya solución depende en gran medida de la experiencia de los expertos.

Problemas actuales de los Sistemas Basados en Casos:

1. Encontrar un modelo de organización e indexación de la memoria de casos que permita un almacenamiento, recuperación y rehúso eficientes.

2. Integrar esa estructura de representación al conocimiento del dominio considerando que puede contener determinada incertidumbre.

1.6 Adaptación en el Razonamiento Basados en Casos.

Una vez que un caso adecuado es recuperado de la Base de Casos, la solución debe de adaptarse, es decir, la solución sugerida por este caso es objeto de una alternativa de reutilización para la solución del problema actual.

Durante este paso, se da una reutilización del conocimiento del caso anterior conocido, para el caso actual, sin embargo este todavía no ha sido solucionado.
Un sistema con RBC debe ser capaz de aplicar el problema actual a la solución descrita para un problema similar recuperado de la Base de Casos. Llamamos a este paso **reutilización.**

La reutilización de los casos tiene mucha importancia ya que con una correcta implementación de esta técnica nos permitirá mejorar la forma en que aprovechamos las

características que tienen los casos y por lo tanto mejorará la forma en que construimos la solución final del problema.

La reutilización de los casos recuperados se fundamenta básicamente en dos aspectos:

• Las diferencias entre el caso recuperado de la Base de Casos y el nuevo caso.

• Que parte del caso recuperado de la Base de Casos puede ser utilizado en el nuevo caso.

La **reutilización** consiste principalmente de adaptación de la solución del caso anterior al caso actual y los métodos tratados. En la reutilización de casos se intentan resolver los problemas envueltos en la adaptación de casos, que son: cuáles aspectos de la situación deben ser adaptados, cuales modificaciones deben ser realizadas para esta adaptación y como controlar este proceso.

Una vez que el caso recuperado de la Base de Casos puede no satisfacer completamente los requisitos dados de la nueva situación, la adaptación se fija en las diferencias entre los casos y aplica reglas de adaptación.

La adaptación adecua la solución del caso más parecido a las condiciones del nuevo caso. Esto es necesario, dado que normalmente los fenómenos o síntomas que se presentan en un diagnóstico, no son idénticos a los ocurridos en los casos anteriores.

Para todo esto existe una serie de métodos que procederemos a describir que nos permitirán mejorar la forma en que reutilizamos un caso existente en la Base de Casos.

Copiar

En los procesos de clasificación más simples, lo que se hace es, en primer lugar, una abstracción de las características del caso que queremos reutilizar olvidándonos de toda aquella información no relevante. Todas estas características son copiadas al nuevo caso que estamos construyendo. Como podemos ver es un método muy sencillo pero hay que tener en cuenta varias cosas como son las diferencias entre los dos casos y que muchas

veces no se pueden copiar directamente algunas características sin antes adaptarlas. Este proceso de adaptación lo vemos a continuación.

Adaptar

Existen dos maneras diferentes para rehusar los casos, o bien rehusar directamente toda la solución del caso en el que nos basamos o bien reutilizar el método que utilizamos para llegar a la solución de ese caso, es decir la manera en que conseguimos llegar a esa solución. Resulta evidente que en la reutilización de un caso, este caso puede no ser el mismo que estamos intentando solucionar actualmente y por lo tanto las soluciones que tenemos en la Base de Casos no serán en la mayoría de los casos soluciones que ofrecer para solucionar el nuevo. Si no se puede aplicar las características del caso que tenemos en la base sobre el nuevo caso entonces estas se adaptan para el nuevo caso mediante una operación de adaptación.

Teniendo en cuenta la naturaleza del problema, se pueden llevar a cabo dos tipos de adaptación en el RBC [2]:

- La **adaptación estructural**: aplica reglas de adaptación directamente a la solución recuperada. Es decir, lo que se reutiliza es la solución del caso previo recuperado. Debe tenerse en cuenta que no siempre es necesario modificar la solución recuperada, sino que podría usarse directamente como solución del problema actual. En lo adelante los métodos que utilizan este tipo de adaptación serán referidos en esta memoria como métodos de sustitución teniendo en cuenta que esta es su actividad fundamental.

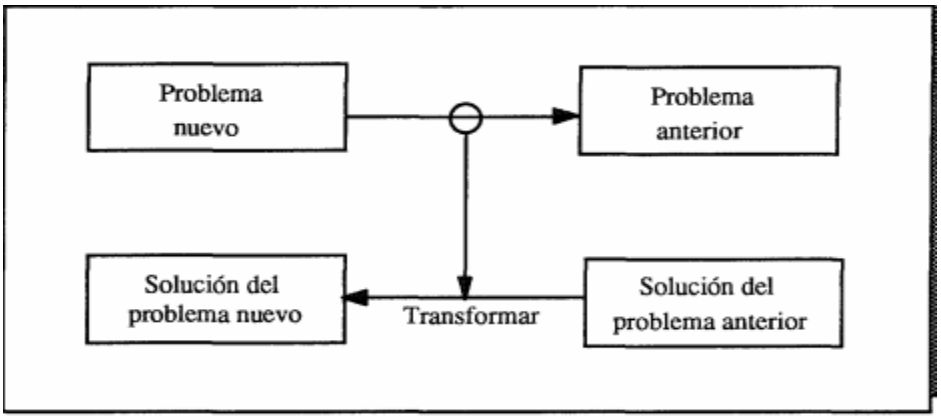

Fig1.4 Analogía de Transformación .

- La **adaptación derivacional**: reutiliza los algoritmos, métodos o reglas para generar la solución pasada para conseguir una nueva solución para el problema actual. Es decir, cuando lo que se reutiliza es el método que construyó la solución del caso previo recuperado. De la misma manera, no siempre es necesario modificar esa secuencia de pasos que se siguió para obtener la solución recuperada, sino que podría usarse sin ninguna modificación. En lo adelante los métodos que utilizan este tipo de adaptación serán descritos en esta memoria como transformación derivacional.

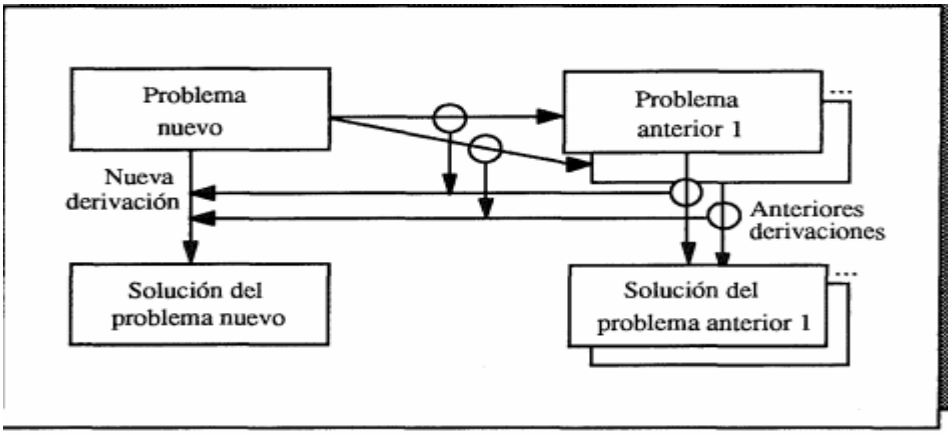

Fig1.5 Analogía de Derivación.

Cuando se recupera un caso, se puede obtener una solución a partir de los múltiples casos, o se puede optar por presentar varias soluciones.

A la hora de elegir entre seguir una u otra estrategia para la reutilización, se debe tener en cuenta el grado de similitud que va a tener el caso recuperado con el caso actual, el número

de características de las que deferirán y si existen reglas que puedan utilizarse para llevar a cabo la reutilización de un modo eficiente.

1.7 Métodos de adaptación.

Existen diferentes métodos que son utilizados en el proceso de adaptación. A continuación se dará una descripción de ellos.

1.7.1 Métodos de Sustitución

Los Métodos de Sustitución dan valores apropiados para la nueva situación dependiendo de los valores de la solución previa. Se puede sustituir solo un componente de la solución, varios de ellos, o todos los componentes de una solución.

1.7.1.1 Reinstanciación

Instancia característica de una solución anterior con las nuevas características. Se usa cuando la similitud entre el caso actual y el recuperado (ó recuperados) es obvia y no hay otros tipos de restricciones o requerimientos impuestos para la solución. Es decir, los casos actuales y recuperados son similares aunque algunos de sus roles tengan valores diferentes, de modo que el proceso a seguir sería tomar la solución recuperada y actualizar los roles que difieran por los nuevos.

Un ejemplo de Reinstanciación lo podemos encontrar en el sistema CHEF [12]. Si tenemos almacenada una receta de pollo al whisky y deseamos una receta de pavo al brandy, el sistema puede ser capaz de recuperar la receta almacenada y darse cuenta que se debe actuar de igual manera, salvo que donde antes era pollo ahora es pavo y donde antes era whisky ahora brandy. Los roles ‘’comida’’ y ‘’licor’’ son los mismos en los dos casos, pero tiene valores diferentes.

Algunos autores consideran que este método debería estar separado del resto pues es más una “copia” que una “adaptación” [1], otros sin embargo si lo consideran como

método de adaptación, pero no lo clasifican dentro del grupo de sustitución [28] y otros en cambio lo engloban dentro de los dos grupos [15].

1.7.1.2 Ajuste de parámetros

Adaptación estructural que compara parámetros del caso extraído con el caso actual, para modificar la solución en la dirección apropiada. Se emplea para interpolar[3] los valores de una solución previa a una nueva solución, es decir, dada una solución previa y un caso nuevo que difiere en algún grado del caso previo, la solución previa se modifica para la extensión en que difieren los dos casos.

Para realizar los cambios en los parámetros de la solución previa, se debe tener en cuenta que grado de diferencia hay con el nuevo caso y en que consisten estas diferencias. Esto se lleva a cabo en dos fases:

- Se comparan los descriptores del problema previo y el nuevo y se extraen las diferencias.

- Se aplican heurísticas [14] de ajuste especializadas a la solución previa para crear la nueva solución. Las mencionadas heurísticas capturarán las relaciones entre características del problema y parámetros de la solución.

JUDGE es un sistema con RBC que utiliza este método de adaptación para solucionar el problema.

1.7.1.3 Búsqueda local

Así como la Reinstanciación sustituía un conjunto completo de roles, hay situaciones en las que solo es necesaria una sustitución de una pequeña parte de una solución previa, para que se adapte perfectamente al nuevo caso.

Se denomina Búsqueda Local al proceso de buscar una jerarquía abstracta (en los entornos de un concepto), algo relativamente similar que pueda ser sustituido. Se suele usar cuando una solución previa es casi correcta para el caso nuevo y puede ajustarse con alguna sustitución menor [4].

La Búsqueda Local debe restringirse puesto que de otro modo sería ineficiente. Además se suele incorporar ciertas guías de movimiento hacia arriba y hacia abajo en la jerarquía.

Una variante de este método es la *Memoria de Consulta,* útil cuando un simple método de búsqueda local no es capaz de predecir adecuadamente las relaciones entre conceptos, lo que conlleva a una búsqueda ineficiente e incluso sin garantías de éxito. Para estos casos, se construyen una descripción parcial de un elemento que sería una buena sustitución e intenta buscarlo. El método particular de búsqueda dependerá de la organización de la memoria (si por ejemplo fuera indexada, se utilizarían los índices como guía).

Otra variante es la *Búsqueda Especializada,* que va un paso más allá de la memoria de consulta: le da a la memoria instrucciones de cómo encontrar el elemento que se necesita. Evidentemente, este método funciona con heurísticas, que deberían ser descritas adecuadamente e incluso pueden agruparse por ciertos criterios.

También existe otra variante más, denominadas *Sustitución Basada en Casos*, la cual simplemente intenta recuperar otro caso que se ajuste lo más posible a esa pequeña parte en que se difiere. Digamos que es una visión recursiva de la sustitución.

1.7.2 Transformación Derivacional.

Los métodos de transformación son una alternativa muy viable, pues generan una nueva solución basándose en las restricciones y características de la solución requerida.

[3] Interpolar, del latino interpolare, alterar, mezclar, cambiar. Poner algo en otra cosa (ver glosario de términos).

Un caso típico en el que se usa, es cuando se intenta buscar una situación pero no se consigue ninguna que satisfaga los requerimientos exactamente. Entonces se utilizará parte o toda la solución recuperada más similar para generar una nueva solución.

1.7.2.1 Transformación de Sentido Común

Este método, la acción que lleva a cabo es transformar la solución usando reglas que utilizan el conocimiento acerca de la importancia relativa y funciones de diferentes componentes.

Básicamente, estas reglas no son más que un conjunto reducido de heurísticas que empleando dicho conocimiento mencionado, determina que transformaciones hay que aplicar a la solución para que funcione adecuadamente en la nueva situación.

Debemos considerar ciertas premisas:

1. Los sistemas deben de ser capaces de identificar el componente que se debe *reparar* de un elemento candidato a ser transformado.
2. Las distintas representaciones deben separar componentes primarios de secundarios.
3. Así mismo, deben mantener relaciones internas entre los elementos de los problemas que se están resolviendo.
4. Las heurísticas juegan un papel de cierta importancia, pues de ellas dependen el definir quién puede ser sustituido.

1.7.2.2 Reparación Guiada por Modelo

Mientras que la Transformación de Sentido Común confía en el conocimiento extraído del funcionamiento regular del mundo, la Reparación Guiada por Modelo es un método de transformación que confía en el conocimiento de conexiones casuales de algún tipo de sistema o situación.

Un ejemplo muy común se da en ciertos tipos de enfermedades; debido a que una elevada presión sanguínea y ciertas arritmias que conocen que son resultado de tener arterias taponadas, la ‘’elevada presión sanguínea ´´ puede ser reemplazada por ´´ arritmia ´´ en una solución previa en la que se trataba con ´´ arritmia ´´ como resultado de ´´ arterias taponadas ´´. Un ejemplo de sistema que utiliza este método de adaptación es el CASEY.

La Reparación Guiada por Modelos es una colección de heurísticas que acceden a modelos casuales para transformar una solución previa de tal modo que se ajuste a la nueva situación.

Para el ejemplo anterior, la heurística utilizada se denomina sustituir- evidencia: si dos estados proveen la misma evidencia para un tercer estado, se puede sustituir entre sí. Y el modelo casual empleado es el que explica el comportamiento cardiaco.

Como el método de Ajuste de Parámetros, este método debe evaluar las diferencias entre la descripción del problema previo y el nuevo. Se trataría de un proceso en 3 pasos:

1. Comparar el problema nuevo y viejo y extraer las diferencias.
2. Evaluar las diferencias usando el modelo casual disponible, caracterizando las diferencias.
3. Para cada diferencia, aplicar a la solución previa la heurística de reparación guiada por el modelo apropiado, creando una nueva solución.

Aunque parezca similar, podemos observar dos diferencias importantes entre el método de Ajustes de Parámetros y la Reparación Guiada por Modelo:

1. El Ajuste de Parámetros sólo puede ajustar valores y cantidades de valores. Sin embargo, la Reparación Guiada por Modelo puede transformar la estructura de una solución previa a reparar algún componente.

2. Las heurísticas de Ajustes de Parámetros están muy especializadas en cierto tipo de parámetros con ciertos tipos de dominios. Aunque parezcan modelar que cosas funcionan, no se apoya en un método sólido. Las heurísticas de Reparación Guiada por Modelo pretenden ser de propósito general y están basadas en nuestro conocimiento casual.

1.8 Otros métodos de Adaptación

Hay casi tantas clasificaciones de métodos como autores. A continuación mencionaremos brevemente otros métodos que quizá lo más interesante es que introduce algún método que difiere de los anteriores [18].

1.8.1 Adaptación Nula

La Adaptación Nula, un método simple directo que aplica cualquier solución que se recupera al problema actual sin adaptarlo.

1.8.2 Reutilización Heurística

La Reutilización Heurística, cuando existen, proveen excelentes guías para llevar a cabo la adaptación. Estas heurísticas guían tres tipos distintos de adaptación:

- Adaptación del Dominio Específico: Referente a reglas de sustitución y transformación específicas para algunos dominios de problemas.

- Modificación Estructural: En este caso se modifica una solución insertando un nuevo elemento en la estructura. A menudo se combinan con modificaciones en la estructura de propósito general.

- Reparación de Propósito General: Recordemos que la reparación es una adaptación que se lleva a cabo cuando la evaluación de la solución ha encontrado algún fallo. La información que se acompaña en el informe del fallo, puede considerarse de

propósito específico y desde luego es una buena guía para saber que debe cambiar en la solución para que funcione.

1.8.3 Reutilización derivacional

En los métodos anteriores, se usa una solución previa para fijar una solución al nuevo problema. Sin embargo, hay ciertas situaciones en las que es más apropiado re-computar una nueva solución o una parte de una solución usando los mismos medios que se usaron para computar la solución previa. [4].

Para poder aplicar este método, los casos almacenados deben mantener información acerca del método usado para resolver el problema, incluyendo una justificación de los problemas usados, submetas consideradas, alternativas generales, caminos de búsqueda fallidos, etc. Cuando se recupera un caso, se reinstancia adecuadamente para la nueva situación y se repite, para el nuevo contexto, el plan que se llevo a cabo.

1.8.4 Abstracción y re-especialización

La abstracción y re-especialización, un método de adaptación estructural general que se usa de una manera básica para lograr las adaptaciones simples y de una manera compleja para generar las nuevas y creativas soluciones. El sistema planificador PLEXUS usa este método.

1.8.5 Adaptación basada en críticas

Adaptación basada en críticas, en que un crítico busca combinaciones de rasgos que pueden causar un problema en una solución. Pretenciosamente, el crítico es consciente de reparaciones para estos problemas.

1.8.6 Adaptación usando algoritmos genéticos.

Es un algoritmo genético. La base de casos forma la población inicial de los genotipos. Primero, el algoritmo recupera casos con un emparejamiento parcial de la Base de Casos con la ayuda de ciertos requerimientos de diseño especificados. A continuación, los casos recuperados se *mapean* en una representación de genotipo para que se apliquen ciertos operadores de mutación. Finalmente, los nuevos genotipos generados se mapean en los correspondientes fenotipos/casos infiriendo valores para los atributos y añadiendo el contexto del nuevo diseño.
La colección de atributos que se emplean para describir un caso viene a ser la colección de genes que forman un genotipo.

1.9 Adaptación en diferentes sistemas.

Para resolver un problema determinado, se han creado muchos sistemas que utilizan el RBC, los cuales realizan adaptaciones para encontrar la solución adecuada al nuevo problema. Dentro de estos sistemas podemos encontrar:

1.9.1 HYPO (Rissland y Ashley 1987, Ashley 1990).

Es un razonador interpretativo basado en casos que trabaja en el dominio de la ley. HYPO toma como entrada una situación legal y como salida crea un argumento para sus clientes legales.

El proceso de razonamiento de HYPO consta de varios pasos:

- Analizar el caso para determinar cuáles son sus factores relevantes de entre todos sus descriptores.
- Recuperar casos que compartan estos factores.
- Separar los casos anteriores en los que apoyan a la defensa y en los que apoyan a la acusación de la nueva situación.

- Seleccionar los casos que comparten el mayor número de características con la nueva situación.
- HYPO crea los argumentos, llamados argumentos de tres capas. Se elige el caso que más ajusta de los de la defensa y el que más ajusta de la acusación. Se buscan diferencias entre estos dos casos y se eligen casos que den soporte a estas diferencias para crear *argumentos* y *contrargumentos*. Este es el principal paso de este proceso, se analizan diferencias y similitudes entre casos antiguos y nuevos y entre varios casos antiguos.
- El análisis realizado en el proceso de argumentación es empleado para justificar la decisión tomada por HYPO.
- Se crean casos hipotéticos que nunca han ocurrido y se usan para validar el análisis realizado y determinar el alcance del análisis.

1.9.2 CHEF (Hammond, 1989)

Es un planificador basado en casos que trabaja en el dominio de recetas de cocina china [12]. CHEF crea las nuevas recetas a partir de otras ya conocidas, como respuesta a unos requisitos (submetas) de platos con ingredientes específicos. Tiene como entrada un conjunto de objetivos (ingredientes) y la salida es una receta que cumpla con los objetivos de entrada. Las recetas nuevas se obtienen seleccionando la receta más parecida y adaptándola en base las discrepancias entre objetivos.

Una vez recuperado el caso más similar, la adaptación se realiza en dos pasos:

- Se reinstancia el plan antiguo.
- Se añaden las restricciones particulares impuestos por los nuevos ingredientes si es necesario.

1.9.3 CASEY (Koton, 1989).

Es un sistema de diagnósticos basado en casos que opera en el dominio de enfermedades del corazón. Tiene como entrada un conjunto de síntomas y características del paciente y la salida es un diagnóstico con una explicación sobre las causas de los síntomas.

CASEY diagnostica los pacientes aplicando heurísticas de adaptación y emparejamiento basado en el modelo. La diagnosis la realiza en dos pasos: primero busca en la memoria casos y utiliza reglas de evidencia basadas en el modelo para determinar cuáles de los casos que ajustan parcialmente son suficientemente similares al nuevo problema para proporcionar una diagnosis precisa. Después aplica reglas de reparación basadas en el modelo (estrategias de adaptación) para adaptar el diagnóstico antiguo a la nueva situación. Las reglas de adaptación están asociadas con las reglas de evidencia; por lo que éstas se almacenan durante el proceso de recuperación para luego ser utilizadas en la adaptación.

1.9.4 PROTOS (Bareiss, 1989)

Implementa clasificación y adquisición de conocimiento basado en casos. El dominio de PROTOS es el de los desordenes auditivos. Para realizar la adaptación utiliza clasificación heurística y técnicas de aprendizaje.

El proceso llevado a cabo por PROTOS es, primeramente, determinar la categoría del nuevo problema buscando, en las categorías de problemas auditivos que conoce, aquella cuyas características importantes coinciden con las características importantes del nuevo problema. A continuación, verifica esta hipótesis intentando ajustar su nuevo caso a ejemplares de la categoría de hipótesis para ver si puede encontrar un buen emparejamiento. Si lo encuentra el proceso acaba.

Si no, utiliza los resultados de este proceso de ajuste para seleccionar una hipótesis mejor. Este proceso se guía por su conocimiento sobre los tipos de errores de clasificación más comunes en este dominio

1.9.5 JULIA (Hinrich, 1992)

Es un sistema de diseño basado en casos que opera en el dominio de planificación de comidas. Como en otros dominios de diseño los problemas se describen en términos de las restricciones que tienen que cumplir, y las soluciones describen la estructura y un artefacto que cumple la mayor cantidad de restricciones posibles. Los problemas son muy extensos y en general no pueden resolverse encontrando un caso anterior que se pueda adaptar para resolver el problema. Habitualmente los problemas se dividen en partes que se resuelven independientemente. Obviamente se debe tener en cuenta las dependencias entre las partes en que se descompone el problema.

Para ello JULIA refuerza el razonamiento basado en casos con un proceso de propagación de restricciones, usando las restricciones como índices del diseño. Las restricciones provienen de diversas fuentes, unas del conocimiento general del artefacto a diseñar (ingredientes incompatibles, por ejemplo) y otras del nuevo problema (los ingredientes que quiere emplear el usuario).

JULIA debe tener un conocimiento general del tipo de artefacto a diseñar almacenándolo en prototipos de objetos, es decir, los prototipos de objetos se corresponden con los diferentes tipos de comida que conoce, por ejemplo, cenas americanas, cenas europeas, comidas de catering, etc. Cada prototipo tiene una estructura específica y relaciones entre sus partes. La adaptación se puede realizar bien por sustitución o bien cambiando la estructura de la solución, en ambos casos guiado por restricciones.

1.9.6 CLAVIER (Hinkle y Toomey, 1994)

Opera en el dominio de carga de un autoclave para la construcción de piezas. Tiene como entrada una lista de piezas a construir y su propiedad y la salida es la disposición de la pieza en el.

Para recuperar contextos similares se utilizan dos tipos de conocimientos:

- Contexto local → distribución de las piezas en la mesa en cuestión.
- Contexto global → las condiciones de la carga: tiempo, temperatura, presión, materiales empleados, etc.

Los fragmentos de los casos –los soportes – están indexados por separado y son susceptibles de ser recuperados.

Es necesario realizar adaptación cuando el caso recuperado incluye piezas que no están en la entrada. No es posible realizar la adaptación considerando sólo la similitud entre piezas aisladas: es necesario tener en cuenta el contexto. Se recuperan casos que coinciden parcialmente con el caso a adaptar –contextos similares – y que incluyan piezas de la entrada. A partir de ellos se sugieren las adaptaciones.

HERRAMIENTAS Y MÉTODOS

2.1 Introducción

En el presente capítulo se explica cómo estará representado el conocimiento en la Base de Casos, la estructura para almacenar los casos, así como la descripción de los algoritmos seleccionados y la herramienta a utilizar para la implementación. Además, recoge los pasos a seguir para su desarrollo y el conocimiento del dominio en general.

2.2 Descripción del problema y solución propuesta.

El RBC es una de las áreas de mayor crecimiento en el campo de los sistemas basados en conocimiento.

Este paradigma es utilizado frecuentemente por los seres humanos para resolver un sinnúmero de situaciones, siendo ésta una de las principales razones de su aceptación en la comunidad de investigadores de IA. La calidad de este tipo de sistemas depende de las experiencias que almacene y de su capacidad para comprender, adaptar, evaluar y reparar nuevos casos.

Hoy en día no existe literatura suficiente sobre técnicas y métodos del RBC, a pesar de la popularidad que el tema ha venido obteniendo.

Uno de los mayores problemas en el RBC se encuentra en el segundo paso de su ciclo, que es la tarea de reutilización [1]. Se debe tener en cuenta que la reutilización se puede identificar por dos maneras, donde en una no se realiza ninguna modificación, mientras que

la otra realiza una adaptación, la cual permite modifica la solución del problema en una dirección apropiada. La reutilización está centrada en dos aspectos:

1. Las diferencias entre el caso pasado y el actual.
2. Qué parte o partes del caso recuperado pueden ser transferidas al nuevo caso.

En algunos casos la tarea de reutilización se reduce a copiar la solución pasada al nuevo caso, pero en otros casos esta solución no puede ser aplicada directamente y tiene que ser adaptada. Si no se realiza una adecuada adaptación, la respuesta al nuevo caso a solucionar no será la correcta. Razones que conducen a realizar un estudio detallado de los métodos de adaptación utilizadas en el RBC y realizar comparaciones para ver su funcionalidad.

Existen muchos sistemas que utilizan RBC, pero cada sistema utiliza un método de adaptación en específico. Como existen diferentes métodos de adaptación y cada uno de ellos realiza este proceso de una forma diferente, se realizó una Base de Casos de recetas de cocina cubana, donde se implementará los métodos seleccionados y se realizará un análisis de los resultados obtenidos,

Este trabajo puede ser un punto de partida para introducir el RBC como ayuda a los desarrolladores de sistemas y contribuir así a lograr software con mayor calidad aprovechando la experiencia acumulada, de manera que se puedan construir soluciones cada vez más complejas y con la rapidez que exige el creciente ritmo de la tecnología de la información.

De gran utilidad serían estos sistemas en empresas desarrolladoras de software, pues la dotarían de una base de ejemplos y ayudarían a futuros desarrolladores a familiarizarse en poco tiempo con las características y políticas definidas en este trabajo.

2.3 Métodos de adaptación seleccionados

De acuerdo al estudio realizado de los diferentes métodos de adaptación utilizados en el RBC, se escogieron para su implementación y prueba los siguientes:

- *Reinstanciación*: Este método adapta soluciones por sustitución de valores dentro de un marco específico y se usa como una estrategia de adaptación principal. Cuando una parte del plan del diseño se encuentra inconsistente, el sistema busca por alternativas en un grupo de casos similares y se escoge el primer similar que encuentra. Entonces el sistema lo substituirá por las inconsistencias. Este proceso garantiza la reducción mínima de inconsistencias en el plan de diseño. Además los métodos de sustitución substituyen los valores convenientes para el nuevo problema de los valores contrarios en el viejo caso. La substitución se puede hacer de muchas diversas maneras. JUDGE substituye nuevos valores para los viejos valores de acuerdo a las diferencias de parámetros entre el viejo caso y el nuevo problema. CLAVIER utiliza la substitución donde un nuevo valor, sugerido por otro caso, substituye un valor inconsistente en la vieja solución.

- *Transformación de Sentido Común*: Este método permite que la vieja solución del caso similar sea transformada en una nueva solución para el nuevo problema. Apoya la reorganización de los elementos de la solución. Dependiendo de la diferencia de los atributos del problema y de los atributos del caso similar, la solución actual es modificada. Así la adaptación requiere conocimiento del dominio en cómo ciertas diferencias en el problema conducen a las diferencias en las soluciones.

- *Adaptación Nula*: Esta adaptación es útil para problemas que involucran el razonamiento complejo pero con una solución simple. Por ejemplo, cuando alguien solicita un préstamo bancario, después de contestar las numerosas preguntas la respuesta final es muy simple: conceda el préstamo, rechace el préstamo, o refiérase la aplicación [17].

2.4 Herramienta para la implementación de los métodos.

Prolog constituye una herramienta extremadamente útil para la construcción de sistemas basados en conocimiento, se hace uso de este lenguaje de programación para la implementación de los métodos de adaptación [9].

Prolog es un lenguaje de programación simple pero poderoso, desarrollado en la Universidad de Marsella como una herramienta práctica para programación lógica. Desde el punto de vista del usuario, la ventaja principal es la facilidad para programar, ya que se pueden escribir rápidamente y con pocos errores, programas claramente leíbles [31].

Se trata de un lenguaje de programación ideado a principios de los años 70 en la universidad de Aix-Marseille por los profesores Alain Colmerauer y Phillipe Roussel. Inicialmente se trataba de un lenguaje totalmente interpretado hasta que, a mediados de los 70, David H.D. Warren desarrolló un compilador capaz de traducir Prolog en un conjunto de instrucciones de una máquina abstracta denominada Warren Abstract Machine, o abreviadamente, WAM. Desde entonces Prolog es un lenguaje semi-interpretado [19].

Prolog es un lenguaje de programación en lógica basado en el Principio de Resolución lineal de Robinson, que trabaja con cláusulas, específicamente con las cláusulas de Horn. Utiliza un sistema de inferencia denominado máquina PROLOG, basada en el principio de resolución de Robinson, el cual permite deducir lógicamente las respuestas a las consultas realizadas en base al conocimiento que previamente se ha incorporado. Utiliza el método de resolución "Top-Down" y una técnica de la lógica formal llamada cálculo de predicados para representar el conocimiento.

La programación lógica tiene sus raíces en el cálculo de predicados, que es una teoría matemática que permite, entre otras cosas, lograr que un computador pueda realizar inferencias, capacidad que es requisito para que un computador sea una "máquina inteligente". La realización del paradigma de la programación lógica es el lenguaje Prolog [31].

Prolog se enmarca en el paradigma de los lenguajes lógicos, lo que lo diferencia enormemente de otros lenguajes más populares tales como Fortran, Pascal, C, Java.

2.4.1 Características.

Un programa en Prolog consiste en la declaración de un conjunto de hechos, que reciben el nombre de base de conocimiento o base de datos y un conjunto de reglas que hacen uso de la base de conocimiento y de otras reglas, este conjunto de reglas recibe el nombre de base de conocimiento procedurales.

Estructura de los programas [10]:

- Los programas consisten en procedimientos.
- Los procedimientos consisten en cláusulas.
- Los programas se ejecutan haciendo preguntas.

Cada una de las cláusulas y hechos son independientes del resto lo que les da a los programas en Prolog un carácter modular. Las variables, como son independientes para cada cláusula, hacen de Prolog un lenguaje distinto al resto de los convencionales, al no ser de naturaleza algorítmica. Es un intento por llegar a un lenguaje declarativo en el cual el usuario formule el conocimiento que quiere representar, a través de conjuntos de hechos y reglas y con lenguaje similar al lenguaje natural.

Prolog es un lenguaje que a pesar de poseer limitaciones tales como: incompletitud en las demostraciones de teoremas, algoritmos de control ligados al orden de las cláusulas y el cuestionamiento general que se hace al formalismo lógico, se ha logrado demostrar su eficacia como herramienta útil a las aplicaciones propias de la IA. El ha sido capaz de enriquecerse con características que permiten el tratamiento de gramáticas, la aceptación de la negación, predicados de entrada y salida y predicados para restringir y orientar mejor las búsquedas a través de meta reglas.

Los programas en Prolog se componen de cláusulas de Horn que constituyen reglas del tipo "modus ponendo ponens", es decir, "Si es verdad el *antecedente*, entonces es verdad el *consecuente*".

En Prolog no existen instrucciones de control. Su ejecución se basa en dos conceptos: la unificación y el *backtracking*.

La unificación es un método interno de manipulación de datos que pasa parámetros, retorna resultados, selecciona y construye estructuras de datos.

Gracias a la unificación, cada *objetivo* determina un subconjunto de cláusulas susceptibles de ser ejecutadas. Cada una de ellas se denomina *punto de elección*. Prolog selecciona el primer punto de elección y sigue ejecutando el programa hasta determinar si el objetivo es verdadero o falso.

En caso de ser falso entra en juego el *backtracking*, que consiste en deshacer todo lo ejecutado situando el programa en el mismo estado en el que estaba justo antes de llegar al punto de elección. Entonces se toma el siguiente punto de elección que estaba pendiente y se repite de nuevo el proceso. Todos los objetivos terminan su ejecución bien en *éxito* ("verdadero"), bien en *fracaso* ("falso").

Las cláusulas de programa y los datos tienen la misma forma. La forma relacional de los procedimientos hace posible definirlos de forma que sean “reversibles”.

Ventajas

- Es rápido de desarrollar.
- Está pensado para la eficiencia.
- Es modular.
- Tiene muy bajo acoplamiento.
- Genera automáticamente un panel de administración.
- Sus bibliotecas hacen gran parte del trabajo.
- Soporta varias bases de datos (MySQL, SQLite, Postgres, MS-SQL)
- Es un lenguaje de propósito general.

2.5 Base de Casos para la prueba de los métodos.

La Base de Casos (BC) contiene las experiencias, ejemplos o casos a partir de los cuales el sistema hace sus inferencias.

Un caso se puede considerar un registro de un problema. La información que se almacena sobre un problema depende tanto del dominio como del propósito para el cual el caso es usado.

Describiremos el funcionamiento básico de los métodos de adaptación del RBC en una BC de Recetas de Cocina Cubana, cuya tarea consiste en construir nuevas recetas a partir de un recetario, en base a un conjunto de requisitos del usuario. La entrada se refiere a diferentes tipos de ingredientes y la salida es una receta única que proporciona una secuencia de pasos a seguir para prepara el plato.

.

2.5.1 Realización de la Base de Casos

La BC está compuesta por 50 casos. La información representada en estos casos son Recetas de Cocina.

Primeramente, su realizó una búsqueda de libros de recetas de comida cubana [7] [22] [23] [24]. Luego se realizó un estudio en cada uno de estos libros, entonces se escogieron diferentes recetas de cocina para guardarlas en la BC.

En los libros de cocina, cada receta pertenece a un grupo característico, por ejemplo, si un individuo desea realizar un plato que sea de arroz con otros ingredientes, busca donde están los arroces y selecciona de esos el que más se ajusta a partir de los otros ingredientes de los que dispone y la selecciona.

Para una mejor organización de los casos en la base y además para facilitar la búsqueda, cada receta pertenece a una clase en específica, es decir, si se tiene organizada un conjunto de recetas y estas pertenecen a las carnes, entonces son del tipo de clase de carnes.

También hay que tener en cuenta el conocimiento del dominio en el que se está trabajando, en este caso como es de receta de cocina, se debe tener conocimiento de los diferentes ingredientes que componen la receta, así como sus diferentes características y cuales ingredientes son sustituibles.

Se realizó un estudio de los ingredientes y se clasificaron en cuanto al grupo de alimento al que pertenecen y en cuanto a su compatibilidad. Como ejemplo podemos mencionar el grupo alimenticio de carnes blancas que son: el pollo, el pavo y el pato. Estos pertenecen a la clase de carnes blancas pero a su vez son compatibles porque se pueden sustituir.

Una vez que la BC se construye, es necesario la existencia de un programa que acceda a ese conocimiento para la inferencia y toma de decisiones en el proceso de solución del problema. Este programa controla el razonamiento y dirige la búsqueda en la BC y es el que generalmente se conoce con el nombre de máquina de inferencia.

La herramienta a utilizar para implementar esta BC es Prolog. El conocimiento descrito es un conjunto de hechos que describe los tipos de ingredientes que se utilizan para elaborar una receta y su clasificación en dependencia al tipo de agrupación al que pertenecen (clases) y el conjunto de recetas (casos) almacenadas en la base para futura reutilización

Para acceder a este conocimiento se define un conjunto de reglas que permiten acceder y realizar consultas a la BC.

Existen otras formas de estructurar la BC, por ejemplo puede ser representada a través de una tabla cuyas columnas son etiquetadas por variables o atributos que representan los rasgos predictores y objetivos, mientras que sus filas representan los casos.

La Tabla 1 representa una base de casos relativa al problema. El espacio E está formado por los casos $C_1, C_2, ..., C_m$, el conjunto de atributos $\{x_1, ..., x_n\}$ son los rasgos predictores, mientras el atributo y_1 representa el rasgo objetivo.

	Rasgos Predictores			Rasgo Objetivo
Caso	x_1	**…**	x_n	y_1
C_1	$x_1(C_1)$	**…**	$x_n(C_1)$	$y_1(C_1)$
…	**…**	**…**	**…**	
C_m	$x_1(C_m)$	**…**	$x_n(C_m)$	$y_1(C_m)$

Tabla 2.1. Tabla que representa una base de casos

En este problema los rasgos predictores son todos los ingredientes principales que se pueden utilizar para desarrollar una receta mientras que el rasgo objetivo es la forma de elaborar la receta.

2.5.2 Estructura de la Base de Casos.

La elección de la estructura de la BC, también llamado modelo de memoria, depende de los siguientes factores:

- La representación usada en la BC.
- El propósito del sistema RBC. Por ejemplo una estructura jerárquica es adecuada para un sistema dedicado a resolver problemas de clasificación. A medida que el número de casos aumenta en la base, un modelo de memoria de búsqueda secuencial (estructura plana), hace aumentar el tiempo de recuperación.
- El número de características usadas para el emparejamiento de casos durante una búsqueda.
- La existencia de casos suficientemente similares como para agruparlos.
- El conocimiento que se tiene sobre el dominio específico. Esto influye en la habilidad para determinar si los casos son similares. Si se dispone de muy poco conocimiento sobre el dominio es posible que la agrupación de casos sea incorrecta.

Luego de escogida las recetas, se define cuál será la estructura para almacenarlas en la BC. Estas recetas están organizadas por: Nombre de la Receta, Ingredientes y Forma de Elaboración donde:

Nombre de la Receta: es el nombre específico para la receta.

Ingredientes: principales recursos alimenticios que van a ser utilizados en el desarrollo de la receta.

Forma de Elaboración: Son los pasos a seguir para componer la receta.

Representación estructural del caso:

Receta ('nombre de la receta', Ingredientes [], Elaboración[]).

Tipos de datos:

Lista

Las listas son una de las estructuras de datos más fundamentales empleadas para almacenar una colección de elementos.

La lista es una estructura de datos muy común en la programación no numérica. Es una secuencia ordenada de elementos que puede tener cualquier longitud. Ordenada significa que el orden de cada elemento es significativo. Un elemento puede ser cualquier término e incluso otra lista. Se representa como una serie de elementos separados por comas y encerrados entre corchetes [13].

La representación de la lista de los ingredientes es: [ing(N,C,U)] donde,

N: nombre del ingrediente

C: cantidad del ingrediente a utilizar

U: unidad de medida, que puede ser:

- ✓ Libras.
- ✓ Kilogramos
- ✓ Gramos
- ✓ Unidades

- ✓ Cucharadas
- ✓ Cucharaditas
- ✓ Taza
- ✓ Litros
- ✓ Etc.

Elaboración []: es una lista que describe como se realiza la receta, es decir, los pasos a seguir para desarrollar la receta.

Como la forma de representar los casos es en un fichero uno a continuación del otro el modelo de memoria utilizado es el de **Memoria Plana** [17].

En la Memoria Plana los casos se almacenan secuencialmente en una lista simple, un arreglo o un fichero. Para lograr una recuperación eficiente, se indexan los casos de la base. En esta memoria los índices se eligen para representar los aspectos importantes del caso y la recuperación involucra comparar las características consultadas con cada caso de la BC
.
Este tipo memoria presenta la ventaja de que añadir nuevos casos resulta muy "barato" (rápido y fácil de implementar). No ocurre así con la recuperación de casos, ya que resulta muy lento cuando el número de casos en la base es alto.
Para este problema el índice que se usará es la clase a la que pertenece la receta, donde a la hora de recuperar el caso se hará por el tipo de la clase a la que es miembro.

2.5.3 Procesos a realizar.

Lo primero que hacemos es realizar una interpretación y determinar las características relevantes de la situación actual.

Las condiciones iniciales, son las características más relevantes del nuevo problema a resolver, en este caso es el ingrediente. Hay que tener en cuenta la cantidad de ingredientes que se desea utilizar para obtener una nueva receta y a la clase a la que pertenece.

Como mencionamos anteriormente, las recetas están agrupadas por clase. Entonces a partir del tipo de clase de la receta que se desea, se recuperarán de la BC aquella que más se asemeje a la nueva receta a resolver, es decir, la que tenga mayor número de características similares al nuevo problema.

El objetivo de la tarea de emparejamiento es devolver un conjunto de casos suficientemente similares al nuevo caso dado un criterio de similitud de algún tipo y la tarea de selección elige entre esos casos el que mejor empareja. El proceso de recuperación no es objetivo central, por tanto se hará una recuperación simple donde se recuperan aquellos casos que tengan elementos similares a la consulta del usuario, es decir, a los datos de entrada.

Luego de obtener la que más ajuste, se realizará el proceso de adaptación con cada método seleccionado. Cada uno de ellos realiza la adaptación de diferentes formas. El tamaño de la entrada o las condiciones iniciales puede variar.

Para emparejar casos basándonos en las similitudes semánticas y la relativa importancia de las características, se necesita un conjunto extensivo del conocimiento general del dominio para dar una explicación de porque dos casos emparejan y como de fuerte es el emparejamiento.

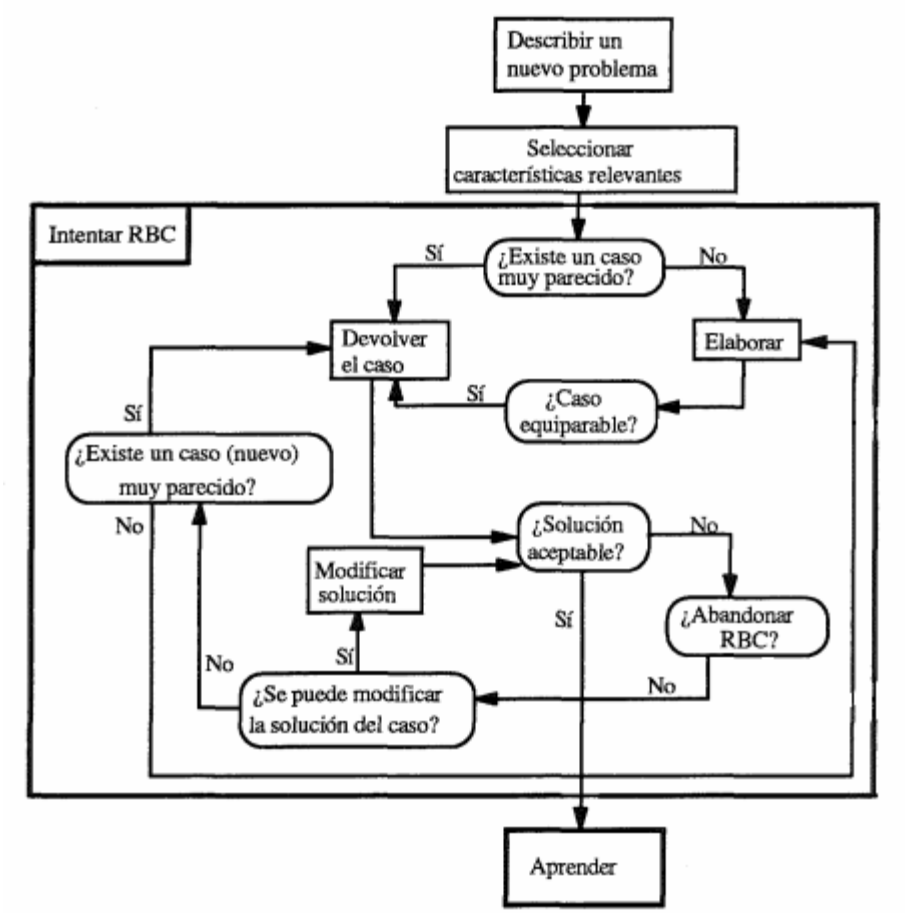

Fig 2.1 Descripción del conjunto de datos.

2.6 Conocimiento del dominio para realizar la adaptación.

Para realizar la adaptación tenemos que tener conocimiento del dominio de aplicación. Como se está trabajando con una BC de recetas de cocina, hay que tener conocimiento de los tipos de alimentos o ingredientes que se utilizan para desarrollar una receta determinada. Cada alimento pertenece a un grupo alimenticio en específico (Ver Anexo1), por tanto cada alimento se organizara en el grupo al cual pertenece.

2.7 Descripción de los algoritmos

A continuación se dará una descripción de los algoritmos de los métodos de adaptación de RBC que serán implementados, así como los pasos a seguir para el desarrollo de los mismos.

2.7.6 Algoritmo de Reinstanciación*.*

La Reinstanciación recupera de la BC aquellos casos que empareja con las condiciones iniciales (CI) del nuevo caso. Luego realiza una indexación del caso conocido (viejo) con la CI del nuevo caso para seleccionar el mejor. Después realiza una adaptación donde se instancia las características que difieren del nuevo caso con el caso recuperado. A continuación se muestra el diagrama de bloques del proceso de Reinstanciación.

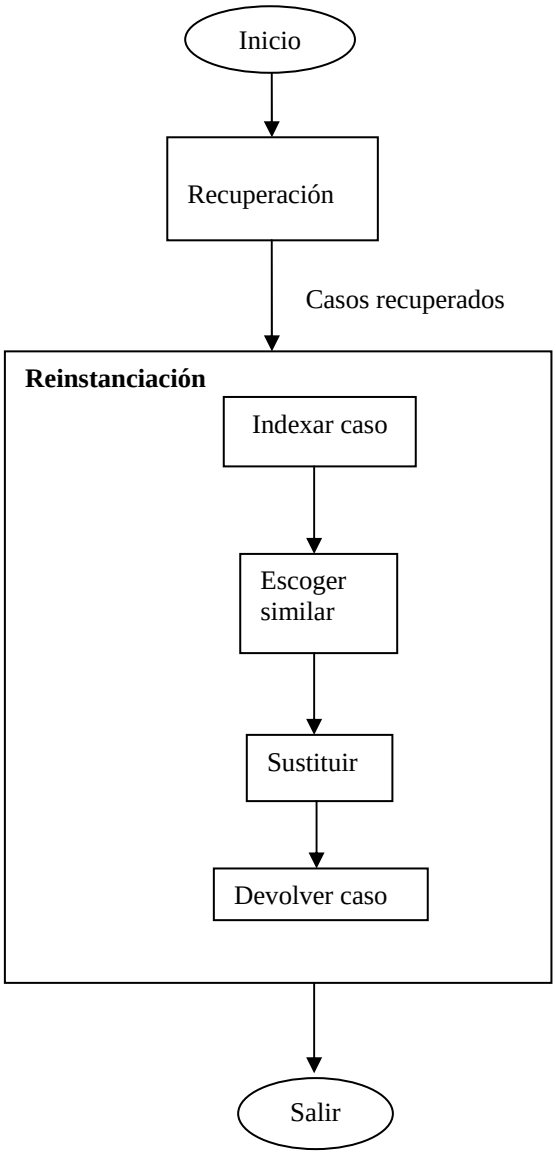

Fig 2.2 Diagrama de bloque del método Reinstanciación.

k: Cantidad de elementos de la BC.

Los casos están estructurados de la siguiente forma:

1. Contiene el identificador del caso.
2. Contiene el tipo de agrupación (Clase) al que pertenecen.
3. Tiene un listado de características (componentes) del caso.

caso[i] : hace referencia al elemento i-ésimo de la BC.

Lista: arreglo [1…..k] de casos

n: nuevocaso→componente→longitud, esta es la cantidad de entrada de la que dependen los algoritmos.

Comp: Lista que guarda por cada caso, la cantidad de componentes similares con el nuevo caso.

```
Reinstanciación (nuevocaso).
inicio
1    cant←0
2   para i←1 hasta k hacer
3          si(nuevocaso→clase = caso[i]→clase) entonces Lista[cant+1]←caso[i]

 4   para i←1 hasta cant hacer
 5    para j←1 hasta nuevocaso→componente→longitud  hacer
 6     if  (nuevocaso→componente[j] = Lista[i]→componente[j]) entonces contador= contador + 1
 7    Comp[i]=contador

 8   Mayor= Comp[1]
 9    pos=1
 10   Para  i←1 hasta nuevocaso→componente→longitud  hacer
 11    If Comp[i]>mayor entonces mayor= Comp[1]  pos=i

 12    devolver modificar (nuevocaso,Lista[pos])
fin

Modificar (nuevocaso,casosimilar)
inicio
1     para i←1 hasta nuevocaso→componente→longitud  hacer
2        si (nuevocaso→componente[i]<>casosimilar→componente[i]) entonces
3              casosimilar→componente[i]← nuevocaso→componente[i]
4    devolver casosimilar
Fin
```

Vamos a pasar a explicar lo que hace el algoritmo.

1. Se inicializa el contador que llevará la cantidad de elementos en la lista de elementos similares al caso nuevo, que esta cantidad en el peor de los casos puede ser igual a k.

2. Se realiza un ciclo que va desde el primer elemento de la bases casos hasta el último elemento, se guarda en una lista aquellos casos que pertenezcan al mismo tipo de agrupación, que serian los casos similares al nuevo caso.

3. Luego se busca por cada caso similar, la cantidad de componentes similares que tiene con el nuevo caso, estos se guardan en la lista Comp, donde la cantidad que esté en la posición i, corresponde con el caso i-ésimo de la lista de casos similares.
4. Teniendo ya guardo en la lista Comp la cantidad similar por cada caso, se pasa a buscar el que mayor grado de similitud tenga, después de recorrer la lista Comp, quedaría en la posición pos, este caso es el que es más similar, y sería el que pasaría a ser modificado.

5. En el método modificar, se realiza un ciclo que recorre las n componentes del caso nuevo, adaptando las componentes del caso nuevo en la solución encontrada.

6. En el método modificar, se realiza un ciclo que recorre las n componentes del caso nuevo, adaptando las componentes del caso nuevo en la solución encontrada.

2.7.2 Algoritmo de Transformación del Sentido Común

La Transformación de Sentido Común recupera de la BC el o los casos más similares al nuevo problema, luego aplica heurísticas de transformación para dar solución al nuevo problema, las cuales son un conjunto de reglas de sentido común que son utilizadas frecuente Este método separa los elementos (ingredientes) primarios de los secundarios.

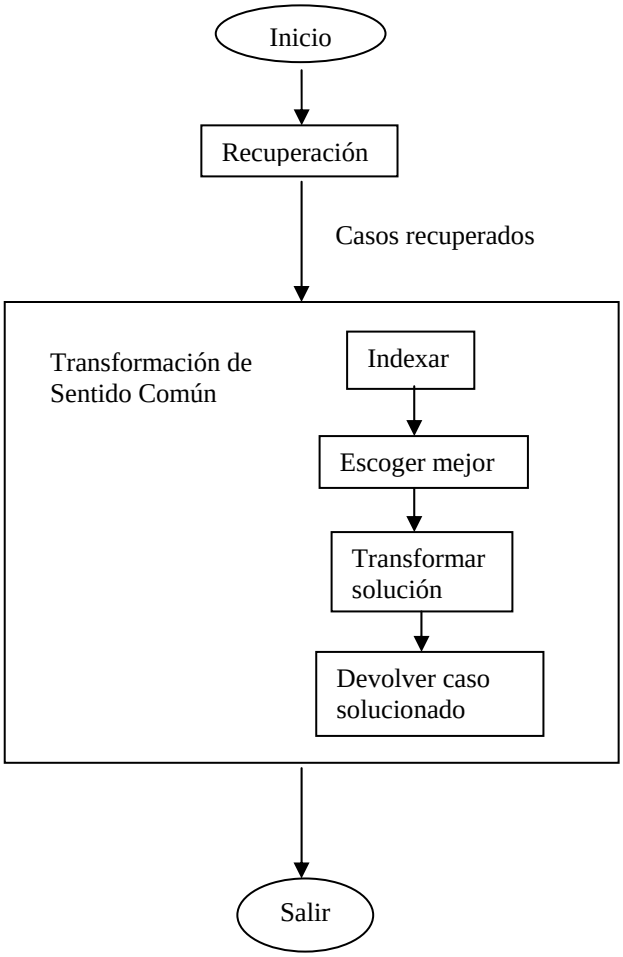

Fig 2.3 Diagrama de bloque del método de Transformación del Sentido Común.

k: Cantidad de elementos de la BC.

Los casos están estructurados de la siguiente forma:

1. Contiene el identificador del caso.
2. Contiene el tipo de agrupación (Clase) al que pertenecen.
3. Tiene un listado de características primarias (componentesprimarios) del caso.
4. Tiene un listado de características secundarias (componentesecundarios) del caso.
5. Diferencia, refleja las diferencias entre el caso analizado y el caso nuevo.

caso[i] : hace referencia al elemento i-ésimo de la BC.

Lista: arreglo [1…..k] de casos

n: nuevocaso→componenteprimarios→longitud, esta es la cantidad de entrada de la que dependen los algoritmos.

```
Transformación (nuevocaso)
 1 cant←0
 2 para i→1 hasta k hacer
 3     si (nuevocaso→clase = caso[i] →clase) entonces Lista[cant+1]←caso[i]

4   para  i←1 hasta cant hacer
5     contador←0
6     para j←1 hasta nuevocaso→componenteprimarios→longitud hacer
7         si (nuevocaso→componenteprimario[j]<>Lista[i] →componenteprimario[j])
8         entonce contador ←contador+1
9     Lista[i] →diferencia←contador+|nuevocaso→componenteprimario→longitud –
10        Lista[i]→componente→longitud|


11  para i→1 hasta cant hacer
12     para j←1 hasta 1 hacer
13        si Lista[i] →diferencia <Lista[j-1] →diferencia entonces
14              valor←Lista[i]
15              Lista[i]←Lista[j-1]
16              Lista[j-1]←valor

17 para i←1 hasta nuevocaso→componenteprimario→longitud hacer
18   si (nuevocaso→componenteprimario[i]<>Lista[1] →componenteprimario[i]) entonce
19      Lista[1] →componenteprimario[i]←nuevocaso→componenteprimario[i]

20 devolver Lista[1] →resultado
```

Este algoritmo expresa lo siguiente.

1. Se inicializa el contador que llevará la cantidad de elementos en la lista de elementos similares al caso nuevo, que esta cantidad en el peor de los casos puede ser igual a k.

2. Se realiza un ciclo que va desde el primer elemento de la bases casos hasta el último elemento, se guarda en una lista aquellos casos que pertenezcan al mismo tipo de agrupación, que serian los casos similares al nuevo caso.

3. En la línea 4, se realiza un ciclo que recorre la lista de casos similares, para buscar las diferencias de cada caso con el caso nuevo, dentro de este ciclo se realiza un ciclo que recorre las n componente comparándolas con el caso analizado en ese momento.

4. De la línea 11 a la 16 se organiza la lista de casos similares de menor a mayor teniendo en cuenta la diferencias, por tanto, en la primera posición quedará el caso que mayor similitud tiene con el caso nuevo

5. A partir de la línea 17, se adapta la solución encontrada con las características del nuevo caso.

2.7.3 Algoritmo de la Adaptación Nula.

La adaptación Nula recupera del conjunto de casos que se encuentra en la base, el más similar al nuevo problema y lo devuelve. Este método no realiza ninguna modificación. Si se desea, el usurario puede realizar la modificación que estime conveniente para solucionar el caso.

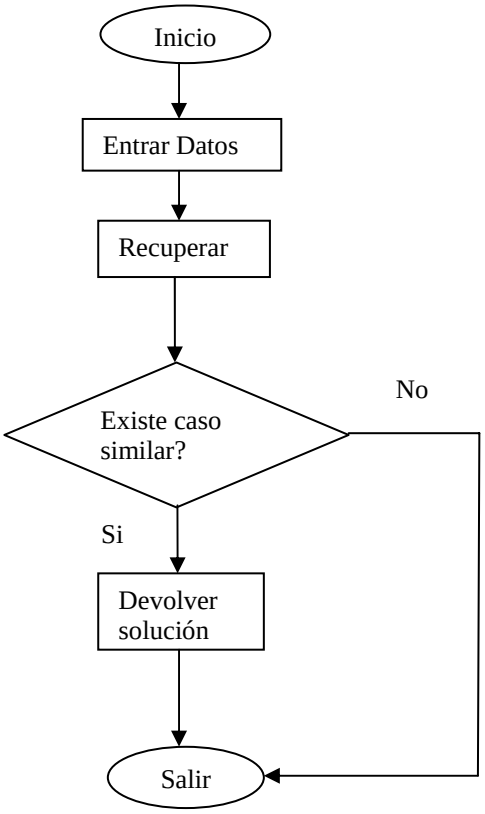

Fig 2.4 Diagrama de bloque del método de Adaptación Nula.

k: Cantidad de elementos de la BC.

Los casos están estructurados de la siguiente forma:

1. Contiene el identificador del caso.
2. Contiene el tipo de agrupación (Clase) al que pertenecen.
3. Tiene un listado de características (componentes) del caso.

caso[i] : hace referencia al elemento i-ésimo de la BC.

Lista: arreglo [1…..k] de casos

n: nuevocaso→componentes→longitud, esta es la cantidad de entrada de la que dependen el algoritmo.

```
Nula (nuevocaso)
1  cant←0
2  para i←1 hasta k hacer
3          si (nuevocaso→clase = caso[i] →clase) entonces Lista[cant+1]←caso[i]

4   devolver Lista
```

Este algoritmo expresa lo siguiente.

1. Se inicializa el contador que llevará la cantidad de elementos en la lista de elementos similares al caso nuevo, que esta cantidad en el peor de los casos puede ser igual a k.

2. Se realiza un ciclo que va desde el primer elemento de la bases casos hasta el último elemento, se guarda en una lista aquellos casos que pertenezcan al mismo tipo de agrupación, que serian los casos similares al nuevo caso.

3. Se devuelven los casos recuperados sin adaptar.

Experimentos y Resultados

3.1 Introducción.

La eficiencia de un algoritmo se puede determinar por la cantidad de recursos informáticos consumidos por él y los resultados que se obtienen en cada algoritmo. Los recursos que habitualmente se contabilizan incluyen la cantidad de memoria y la cantidad de tiempo de cómputo requerido por el algoritmo.

3.2 Característica a Evaluar.

Las características básicas de un algoritmo es que sea correcto, es decir, que produzca el resultado deseado en tiempo finito. La eficiencia de los mismos se define por la cantidad de recursos de cómputo que requiere, es decir, cual es su tiempo de ejecución y qué cantidad de memoria utiliza. Un algoritmo será tanto más eficiente cuanto menos recurso consuma: tiempo y espacio de memoria. Su coste en tiempo depende del tipo de operaciones que realiza y el coste específico de estas operaciones.

Luego de haber implementado las técnicas de adaptación se prosigue a realizar las comparaciones. Estas pueden ser:

- El tiempo que tarda en la corrida del programa, dependiendo de:
  - El tamaño de la entrada.
- La complejidad Algorítmica.
- Espacio en memoria que utiliza.
- La respuesta que arroja cada uno de los algoritmos.

Cada técnica de adaptación tiene sus características. Algunas son más complejas que otras, por lo que hay que tener presente el grado de complejidad de cada una.

En este trabajo estudiaremos el coste en el caso peor de los algoritmos porque:

1. Proporciona garantías sobre el coste del algoritmo ya que en ningún caso excedería el coste en caso peor.
2. Es más fácil de calcular que el coste en el caso promedio

Definimos el *tiempo de ejecución en el caso peor* de un algoritmo como el máximo de los tiempos de ejecución de ese algoritmo para todas las posibles entradas de tamaño *n.*

3.3 Experimento 1 Complejidad Algorítmica

En este experimento la característica a evaluar será la complejidad algorítmica. Para ello hay que determinar el tiempo requerido por cada operación elemental y multiplicarlo por el número de veces que la operación es usada al implementar el algoritmo.

Se calculó el orden de complejidad de los diferentes algoritmos de adaptación y se determinaron los siguientes:

1. Reinstanciación K*N.
2. Transformación de Sentido Común K^2
3. Nula K

Según el tiempo de ejecución podemos determinar que la adaptación Nula es el que menor tiempo utiliza, pero tiene la desventaja que no adapta la solución encontrada al nuevo caso, por tanto, creemos que no sería un buen algoritmo de adaptación, pues no arrojaría una solución acorde al problema planteado.

Como el orden de complejidad del método de Reinstanciación es menor que el de Transformación de Sentido Común, se concluye que el mejor método es el de Reinstanciación.

3.4 Experimento 2. Tiempo de Ejecución

Usaremos el término **tiempo de ejecución** (o simplemente **tiempo**) para referirnos al tiempo de cómputo requerido por un algoritmo en función de n. Denominaremos T(n) al tiempo de ejecución de una algoritmo de tamaño n, donde N es el número de componentes sobre los que se va a ejecutar el algoritmo. Esta función se puede medir físicamente (ejecutando el programa, reloj en mano).

En el presente experimento se calculará el tiempo de la corrida de los algoritmos usando el predicado **ms** en Prolog que me permite determinar el tiempo en segundo al correr cada uno de los algoritmos de adaptación.

La ejecuciones de los algoritmos se realizaron en una Pentium 4, Intel (R) Core (TM)2 Duo CPU T7500, 2.26 GHz (2 CPUs), 2GB de memoria RAM, compilador WIN-PROLOG 3.333.

Denominaremos T_{AN} (n) al tiempo que demora en dar respuesta el algoritmo de Adaptación Nula, T_{R} (n) al tiempo del algoritmo de Reinstanciación y T_{SC} (n) al tiempo del algoritmo de Transformación de Sentido Común.

Algoritmo	Tiempo(s)
T_{AN} (1)	T =0
T_{R} (1)	T=1,3
T_{SC} (1)	T=1,5

Tabla 3.1 Tiempo de ejecución para n=1

Algoritmo	Tiempo(s)
T_{AN} (2)	T =0
T_{R} (2)	T=0,05
T_{SC} (2)	T=2,6

Tabla 3.2 Tiempo de ejecución para n=2

Como podemos observar la adaptación Nula siempre tendrá tiempo cero porque solo devolverá aquellos casos que más se ajusten a las condiciones de entrada del nuevo caso sin ninguna modificación. A pesar que la Transformación de Sentido Común solo busca por los elementos primarios, su tiempo de ejecución es mayor pues esta tiene que realizar más operaciones elementales para devolver la lista de los elementos primarios y la de elaboración modificados, más la lista de elementos secundarios sin modificar. Entonces el método de Reinstanciación solo devuelve la lista de elementos y la de elaboración ya modificados realizando menos operaciones elementales, entonces tiene menor tiempo de ejecución.

3.5 Experimento 3 Espacio en Memoria

Otras de las características por las que se puede medir la eficiencia de un algoritmo, es en cuanto al espacio de memoria utilizado. Se puede determinar entonces que la Reinstanciación y la Transformación de sentido Común tiene igual espacio de memoria porque el primero tienen un campo donde se pasará a guardar las diferencia de cada caso con el nuevo caso, mientras que el segundo tiene una lista donde se guardan la cantidad de casos similares. Sin embargo la adaptación Nula posee menor espacio en memoria pues solo devuelve las casos similares pero sin modificación

3.6 Experimento 4 Respuestas Obtenidas

En el siguiente experimento se determinará la eficiencia de los algoritmos en cuanto a la respuesta arrojada por cada uno. Para ello se realizará tres pruebas donde se mostrarán los resultados de cada uno de los algoritmos. Para cada prueba primero se pasara los ingredientes a utilizar en la nueva receta y la clase a la que pertenece, luego a partir de los datos de entrada, en este caso es una lista de ingredientes, se buscará en la base aquellas que más se ajusten a estas condiciones y en caso de haber modificaciones, las realizara de forma adecuada.

Es decir, si se quiere elaborar una nueva receta de arroz con jamón, las condiciones de entrada es una lista con estos ingredientes [arroz, jamón] más la clase a la que pertenece, en este caso es a la clase de arroces, luego se busca en la BC todas las recetas que ajusten con la condiciones de entrada. Si son varias, escoge la mejor, de encontrar una sola, solo devolverá esta, luego realiza modificaciones entre la receta recuperada y la nueva receta para obtener la respuesta adecuada, en este caso lo que se desea conocer es la forma de elaborar la nueva receta a partir de la recuperada. Si por ejemplo, la receta que mejor empareja es de arroz con salchicha, se modificará aquellos componentes que difieren.

En el método de Adaptación Nula y en el de Reinstanciación se pasa una lista de ingredientes, sin embargo en el método de Transformación de Sentido Común se separan los primarios de los secundarios y siempre se buscará por los primarios.

Prueba #1

Como dato de entrada se pasara una lista con dos elementos, en este caso dos ingredientes [arroz, pavo].

Como es una receta de arroz, la clase a la que pertenece es de tipo arroces.

La salida debe ser una lista de ingredientes I y la elaboración E de la receta.

Si se desea se puede extraer los ingredientes donde solo se devuelve una lista L de ingredientes sin la cantidad a utilizar ni la unidad de medida.

En la BC la receta que más ajusta con los datos de entrada es la receta de “arroz con pollo”.

Respuesta:

Adaptación Nula

I=[ing(arroz,2,tazas),ing(pollo,5,piezas),ing(caldo_pollo,3,tazas),ing(sofrito_criollo,1 / 4,taza),ing(aceite,2,cucharadas),ing(vino_seco,1 /4,taza),
ing(sal,1,cucharada)] ,

L = [arroz, pollo, caldo_pollo, sofrito_criollo, aceite, vino_seco, sal]

E = [sazona el pollo con sal dóralo en la sartén con el aceite añade el sofrito_criollo a la olla agregue el arroz incorpora el caldo la sal introduce las porciones de pollo cóselo a presión durante 8 min cuando la olla pierda la presión destápela perfume con vino seco tápela por 2 min más ábrela remueva el arroz];

Respuesta:

Reinstanciación

I=[ing(arroz,2,tazas),ing(pavo,5,piezas),ing(caldo_pollo,3,tazas),ing(sofrito_criollo,1 / 4,taza),ing(aceite,2,cucharadas),ing(vino_seco,1/4,taza),
ing(sal,1,cucharada)] ,

I = [arroz, pavo, caldo_pollo, sofrito_criollo, aceite, vino_seco, sal],

E = [sazona el pavo con sal dóralo en la sartén con el aceite añade el sofrito_criollo a la olla agregue el arroz incorpora el caldo la sal introduce las porciones de pavo cóselo a presión durante 8 min cuando la olla pierda la presión destápela perfume con vino seco tápela por 2 min más ábrela remueva el arroz];

Respuesta
Transformación de Sentido Común

P = [ingp(arroz,2,tazas),ingp(pavo,5,piezas)] ,

S=[ings(caldo_pollo,3,tazas),ings(sofrito_criollo,1/4,taza),ings(aceite,2,cucharadas),ings(vino_seco,1 / 4,taza),ings(sal,1,cucharada)] ,

P = [arroz,pavo] ,
S = [caldo_pollo,sofrito_criollo,aceite,vino_seco,sal] ,

E = [sazona el pavo con sal dóralo en la sartén con el aceite añade el sofrito_criollo a la olla agregue el arroz incorpora el caldo la sal introduce las porciones de pavo cóselo a presión durante 8 min cuando la olla pierda la presión destápela perfume con vino seco tápela por 2 min más ábrela remueva el arroz];

Prueba #2

Se pasar una lista con elementos, en este caso los ingredientes [huevos, salchicha].
Se desea obtener una nueva receta a partir de los ingredientes de entrada. La nueva receta es una tortilla de huevo con salchicha, por tanto la clase a la que pertenece es de tortilla.
En la BC la receta que más ajusta con los datos de entrada es la receta de “tortilla de jamón”.

Respuesta:
Adaptación Nula

I=[ing(huevos,2,unidades),ing(jamon,2,rodajas),ing(aceite,1,cucharada),ing(cebolla,1 / 2,unidad),ing(sal,1,pizca)] ,
L = [huevos, jamón, aceite, cebolla, sal]

E = [corte las rodajas de jamón en cuadritos corte la cebolla en rodajas bata los huevos agregue el jamón la cebolla verter en la sartén el aceite cuando esté caliente agregue la mezcla voltee por ambos lados hasta que se cocine bien servir]

Respuesta:

Reinstanciación

I=[ing(huevos,2,unidades),ing(salchicha,2,rodajas),ing(aceite,1,cucharada),ing(cebolla,1 / 2,unidad),ing(sal,1,pizca)] ,
I = [huevos, salchicha, aceite, cebolla, sal] ,

E = [corte las rodajas de salchicha en cuadritos corte la cebolla en rodajas bata los huevos agregue el jamón la cebolla verter en la sartén el aceite cuando esté caliente agregue la mezcla voltee por ambos lados hasta que se cocine bien servir]

Respuesta
Transformación de Sentido Común

P = [ingp(huevos,2,unidades),ingp(salchicha,2,rodajas)] ,
S = [ings(aceite,1,cucharada),ings(cebolla,1 / 2,unidad),ings(sal,1,pizca)] ,

EP = [huevos, salchicha] ,
ES = [aceite, cebolla, sal] ,

E = [corte las rodajas de salchicha en cuadritos corte la cebolla en rodajas bata los huevos agregue el jamón la cebolla verter en la sartén el aceite cuando esté caliente agregue la mezcla voltee por ambos lados hasta que se cocine bien servir];

Prueba #3

Como dato de entrada se pasara la lista de ingredientes [dorado, tomate].
En este caso la receta es de pescado, por tanto pertenece a la clase de pescado.
En la BC la receta que más ajusta con los datos de la entrada es la receta de “pargo con tomate”.

Respuesta:

Adaptación Nula

I=[ing(pargo,10,filetes),ing(tomate,1,taza),ing(aceite,1,taza),ing(sal,1,cucharadita),ing(jugo_limon,1 /2,taza),ing(pimienta,1,cucharadita),ing(huevos,2,unidades)] ,
E = [sazonar el pargo con el jugo_limon sal pimienta rallar el pan pasar los filetes por los huevos batidos pan rallado freír en aceite bien caliente hasta que se dore dar vuelta para que se dore por ambos lados servir con tomate en ensalada]

Respuesta:

Reinstanciación

I=[ing(dorado,10,filetes),ing(tomate,1,taza),ing(aceite,1,taza),ing(sal,1,cucharadita),ing(jugo_limon,1 /,taza),ing(pimienta,1,cucharadita),ing(huevos,2,unidades)] ,
I = [dorado,aceite, sal, jugo_limon, pimienta, huevos] ,
E = [sazonar el dorado con el jugo_limon sal pimienta rallar el pan pasar los filetes por los huevos batidos pan rallado freír en aceite bien caliente hasta que se dore dar vuelta para que se dore por ambos lados servir con tomate en ensalada];

Respuesta
Transformación de Sentido Común

P = [ingp(dorado,10,filetes),ingp(tomate,1,taza)] ,
S=[ings(aceite,1,taza),ings(sal,1,cucharadita),ings(jugo_limon,1/2,taza),ings(pimienta,1,cucharadita),ings(huevos,2,unidades)] ,

P = [dorado,tomate] ,
S = [aceite,sal,jugo_limon,pimienta,huevos] ,

E = [sazonar el pargo con el jugo_limon sal pimienta rallar el pan pasar los filetes por los huevos batidos pan rallado freír en aceite bien caliente hasta que se dore dar vuelta para que se dore por ambos lados servir con tomate en ensalada]

Como se puede apreciar la Adaptación Nula no realiza ninguna modificación aunque si encuentra la receta que mejor a justa con las condiciones de entrada. Sin embrago la Reintanciación y la Transformación de Sentido Común si realizan una buena recuperación y adaptación a partir de las condiciones iniciales que se plantean, solo hay que tener presente que una busca en toda la lista de ingredientes pero la otra solo realiza la búsqueda a partir de los componentes primarios.

3.7 Resultados

En cuanto a los resultados que arroja cada algoritmo, podemos concluir que:

- La adaptación Nula tiene menor espacio en memoria, menor orden de complejidad y en la corrida del método T(n)=0, pero como se expresó anteriormente, no adapta el caso nuevo al caso similar encontrado.
- La Reintanciación y la Transformación de Sentido Común poseen el mismo espacio en memoria y adaptan el caso recuperado para darle solución al nuevo problema.
- La Transformación de Sentido Común, a pesar que separa los primarios de los secundarios, posee mayor tiempo de ejecución que la Reintanciación.
- En cuanto al orden de complejidad la Reinstanciación posee menor orden que la Transformación de Sentido

  A partir de estos resultados podemos concluir que el mejor método de adaptación es el de la Reinstanciación.

Conclusiones

El Razonamiento Basado en Casos puede resolver problemas de diseño, planificación, clasificación, interpretación, planeamiento, diagnosis, explicación, análisis, mediación, argumentación, proyección de efectos, monitoreo, razonamiento creativo, entre otros. Por tanto, el estudio de los aspectos fundamentales de esta rama de la Inteligencia Artificial brinda un útil conocimiento para que los analistas de sistemas puedan desarrollar software con mejor calidad y eficiencia.

El presente trabajo de diploma contribuye a dar solución a la situación existente referida a la toma de decisión en los métodos de adaptación a utilizar en determinados problemas, a través de la comparación de los algoritmos definidos a partir de los algunos métodos de adaptación.

La investigación arrojó la siguiente conclusión:

- Se hizo una búsqueda de los diferente métodos de adaptación, algoritmizando algunos de estos.
- Teniendo en cuenta las características de comparación, el método de Reinstanciación es una buena opción a la hora de escoger un algoritmo de adaptación.
- Con el conocimiento recogido en el trabajo, el centro y la carrera de informática cuenta con una información importante a consultar por futuros investigadores.

Recomendaciones

- Proponer más trabajos de diplomas relacionados con esta temática para dar seguimiento a esta investigación.

Bibliografía

[1] AAMODT, A., Plaza, E. *Case-Based Reasoning: Foundation Issues,Methodological Variations, and System Approaches*. AI Communications, vol 7, no 1, 1994. 39-52p Disponible en : http://www.iiia.csic.es/People/enric/AICom.pdf (17/2/2009)

[2] BALÁZS, M., IMRE, S., LAJOS, K. *Adaptation Methods in Case-Based Reasoning.* Technical University of Budapest.1998.5p. Disponible en: http://www.manuf.bme.hu/~miko/mc99.pdf (20/2/2009).

[3] BELLO, R. *Modelos y programas para la solución de problemas usando Razonamiento Basado en Casos.* Universidad Central "Marta Abreu" Academia de Ciencias. Las Villas. República de Cuba. 2004. Disponible en: http://www.academiaciencias.cu/paginas/presentacion/reconocimientos/premios.asp?idp=981&nsecc=Ciencias%20T%C3%83%C2%A9cnicas.(4/5/2009).

[4] CARILLO, J.*A. Introducción al Razonamiento Basado en Casos (CBR)*. Universidad de Valladolid. mayo 2007. 35p. Disponible en: http://www.infor.uva.es/~calonso/IAII/Aprendizaje/TrabajoAlumnos/RBCmemoria.pdf (20/5/2008).

[5] CUADRADOS RODRIGUEZ, S., CURBELO HERNÁNDEZ, H., GONZÁLEZ RODRIGUEZ, E., ***et al.*** *Uso del Razonamiento Basado en Casos combinado con técnicas estadísticas para el diagnóstico de la Hipertensión Arterial.* Universidad Central "Marta Abreu". Las Villas. Cuba.2008 9p. Disponible en: http://cencomed.sld.cu/hta2008/trabajoshta2008/carteles/epidem/EPID18/EPID18.doc (12/5/09)

[6] DELGADO DAPENA, M.D. *Definición del modelo de negocio y del dominio utilizando Razonamiento Basado en Casos.*

[7]EUGUES C., GLADIS E. *Para tu Deleite: Recetario de Cocina.* Cuba 2Ed.febrero. 2004. 191p.

[8] FERNÁNDEZ , R., PERDOMO, G. *Razonamiento Basado en Caso en Ciencias Médicas sobre Plataforma* Web. Revista Cubana de Informática Médica. 2007. Disponible en: http://www.informatica2007.sld.cu (15/4/2009).

[9] [10] FUTCH, E. *Programación en Prolog para Inteligencia Artificial,* http://maestros.unitec.edu/~efutch/concapanxxiii-prolog.pdf*** (25/5/2009).

[10]FUTCH, E*. Programación en Prolog para Inteligencia Artificial.* http://maestrso.united.edu (20/1/2009)

[11] GÁLVEZ LIO, D. *Sistemas Basados en el Conocimiento.* Universidad Central "Marta Abreu". Las Villas. Cuba. 2006.38-59p.

[12]Hammond, K. J. *Case-based planning: A frame-work for planning from experience*. Cognitive Science. 1990. CHEF. Diponible en: http://citeseer.ist.psu.edu/50089.html

[13] HEILEMAN, G. *Estructuras de datos, Algoritmos y Programación Orientada a Objetos.* La Habana. Cuba. 2003. 19-34p.

[14] *Heurísticas.* http://es.wikipedia.org/wiki/Heur%C3%ADstica (26/6/ 2009).

[15]KOLODNER, J. *Case-Based Reasoning*. San Mateo: Morgan Kaufmann, 1993.

[16] *Logia programming Associates Ltd, LPA case Based Reasoning Toolkit* http://www.lpa.co.uk/cbr.htm (5/11/2008).

[17] LOUIS, E., FRENZEL, JR. *Crash Course in Artificial Intelligence and Expert System.*

[18] LOZANO, L., FERNÄNDEZ, J. *Razonamiento Basado en Casos: Una Visión General.* Universidad de Valladolid, 59p.

[19] MONTES DE OCA, JOEL.A. *Modelo de un Sistema Basado en Casos para el diagnóstico del Síndrome de Maloclusión.* Universidad Médica de La Habana. Cuba. 2006.

[20] PADRON SOROA. S., MORELL PÉREZ, C. *Aplicación de la Inteligencia Artificial como ayuda en la elaboración de Tecnologías de Maquinado de Piezas Simétrico Rotativas*. XIV Forum de Ciencia y Técnica. La Habana. Cuba.2007. Disponible en:

http://www.cubaindustria.cu/PiezasRepuesto/Ponencias14forum/0501088. (26/6/2009).

[21] *Prolog. http://es.wikipedia.org (5/3/ 2008).*

[22] PUPO, N. *Especial Mujeres: Comida Cubana, Aliños, Conservas y Plantas Aromáticas*. Cuba. Editorial de la Mujer. Otubre, 2005. 32p.

[23] PUPO, N. *Especial Mujeres: Comida Cubana, Arroces y Pastas.* Cuba. Editorial de la Muje. Octubre, 2005. 32p.

[24] PUPO, N. *Especial Mujeres: Comida Cubana, Croquetas, Frituras y Paltos Fríos*. Cuba. Editorial de la Mujer. Octubre 2005. 32p.

[25] *Razonamiento Basado en Casos.* Disponible en: http://es.wikipedia.org/wiki/Razonamiento_basado_en_casos (26/4/2009).

[26] RIESBECK, C., SCHANK ,R. *Inside Case-based Reasoning*. Northvale, NJ: Erlbaum. 1989.

[27] RODRÍGUEZ SARABIA, Y., FERNÁNDEZ SÁNCHEZ, K.L., GARCÍA LORTENZO, M.M., ***et al***. *Sistema Experto para el diagnóstico y tratamiento del Embarazo Ectópico.* 1er Simpósium Cubano de Inteligencia Artificial.2004.1p. http://www.bibliociencias.cu/gsdl/collect/eventos/index/assoc/HASH903d.dir/doc.doc (5/6/2009).

[28] SANKAR, K., SIMON, P; *Foundations of soft Case-Based Reasoning*. Editorial John Wiley & Sons. 2004.

[29] SCHANK, R Dynamic Memory: *A Theory of Learning in Computers and People.* Cambridge University Press. New York. 1982

[30] SOTO BERUMEN, L.H., MOYA RODRIGUEZ, J.L., CHAGOYEN MÉNDEZ, C.A. *Aplicaciones de los Sistemas Basados en el Conocimiento al Diseño de Transmisiones por Tornillo sin fin.* Universidad de Zacateca. Universidad de Las Villas. México-Cuba. http://eventos.fim.uclv.edu.cu/comec/cd/ponen/c2/c2.74.pdf (26/6/2009).

[31]*Temas Generales de Prolog.1997.* http://www.monografias.com/trabajos5/prolog/prolog.shtml (24/5/2009)

[32]WATSON, I., MARIR, F. *Case-Based Reasoning: A Review.* University of Auckland. New Zeland. 2000.http://www.ai.cbr.org (13/11/2008).

Glosario de Términos.

Adaptar: Modificación destinada a darle una forma diferente a la original. Acomodar, ajustar algo a otra cosa.

Algoritmos: Conjunto ordenado y finito de operaciones que permitan hallar la solución de un problema.

Constreñimientos: Apremio y compulsión que se da a alguien para que ejecute algo.

Heurística: Manera de buscar la solución de un problema mediante métodos. || 2. Se denomina heurística a la capacidad de un sistema para realizar de forma inmediata innovaciones positivas para sus fines. La capacidad heurística es un rasgo característico de los humanos, desde cuyo punto de vista puede describirse como *el arte y la ciencia del descubrimiento y de la invención* o de resolver problemas mediante la creatividad y el pensamiento lateral o pensamiento divergente.

Indexar: Registrar ordenadamente datos e informaciones para elaborar su índice.

Inferir: Sacar una consecuencia o deducir algo de otra cosa.

Interpolar: Calcular el valor aproximado de una magnitud en un intervalo cuando se conocen algunos de los valores que toma a uno y otro lado de dicho intervalo.

Mapear: Representa las partes de un todo.

Métodos: Procedimiento que se sigue en las ciencias para hallar la verdad y enseñarla.

Modificar: Transformar o cambiar algo de sus accidentes.

Regla: Conjunto de operaciones que debe llevarse a cabo para realizar una inferencia o deducción correcta.

Transformar: Hacer cambiar de forma a alguien o algo.

Anexo

Conocimiento del dominio para realizar la adaptación.

Alimento	
• Carne Roja	• Carne Blanca
• Viseras	• Bebida
• Granos	• Fruta
• Vegetal	• Condimentos
• Viandas	• Jugo
• Azúcar	• Huevo
• Vino	• Agua
• Salsa	• Pan
• Queso	• Pescado
• Embutidos	• Marisco
• Picadillo	• Pastas
• Grasa	• Arroz
• Caldo	• Cítrico

Tabla Anexo 1.1 Conocimiento del dominio general.

Alimento	Ingredientes

Carne Roja	• Res • Ovejo • Cerdo • Chivo • Conejo • Carey • Visera

Tabla Anexo 1.2 Grupo de ingredientes de Carne Roja.

Alimento	Ingredientes
Carne blanca	• Pollo • Pato • Pavo • Pescado • Marisco

Tabla Anexo 1.3 Grupo de ingredientes de Carne Blanca.

Alimento	Ingredientes
Visera	• Hígado • Corazón • Pechuga • Riñones • Molleja

Tabla Anexo 1.4 Grupo de ingredientes de Visera.

Alimento	Ingredientes
Carne	• Carne roja • Carne blanca • Visera

Tabla Anexo 1.5 Grupo de ingredientes de Carnes

Alimento	Ingredientes
Azúcar	• Azúcar blanca • Azúcar parda

Tabla Anexo 1.6 Grupo de ingredientes de Azúcar.

Alimento	Ingredientes
Arroz	• Arroz • Arroz integral

Tabla Anexo 1.7 Grupo de ingredientes de Arroz.

Alimento	Ingredientes
Bebida	• Whisky • Cerveza • Ron • Aguardiente • Te • Chocolate • Leche • Yogurt • Vino tinto • Vino blanco • Jugo

Tabla Anexo 1.8 Grupo de ingredientes de Bebida.

Alimento	Ingredientes
Caldo	• Caldo de pollo • Caldo de pescado • Caldo de vegetales • Caldo de frijol colorado • Caldo de frijol negro • Caldo blanco

Tabla Anexo 1.9 Grupo de ingredientes de Caldo.

Alimento	Ingredientes
Cítrico	• Limón • Naranja • Naranja agria

Tabla Anexo 1.10 Grupo de ingredientes de Cítrico

Alimento	Ingredientes
Conservas	• Pepino encurtido

Tabla Anexo 1.11 Grupo de ingredientes de Conservas.

Alimento	Ingredientes
Embutido	• Chorizo • Salchicha • Perro caliente • Jamón • Tocino • Mortadela • Jamonada

Tabla Anexo 1.12 Grupo de ingredientes de Embutidos.

Alimento	Ingredientes
Fruta	• Fruta bomba • Piña • Cítrico

Tabla Anexo 1.13 Grupo de ingredientes de Fruta.

Alimento	Ingredientes
Grano	• Frijol negro • Frijol blanco • Frijol colorado • Maíz • Chícharo • Garbanzo • Lenteja • Arroz

Tabla Anexo 1.14 Grupo de ingredientes de Grano.

Alimento	Ingredientes
Grasa	• Aceite de oliva • Aceite vegetal • Aceite • Manteca

Tabla de Anexo 1.15 Grupo de alimento de Grasa

Alimento	Ingredientes
Harina	• Harina de castilla • Harina de yuca • Harina maíz • Harina de trigo • Maicena

Tabla Anexo 1.16 Grupo de ingredientes de Harina.

Alimento	Ingredientes
Jugo	• Jugo de tomate • Jugo de piña • Jugo de zanahoria • Jugo de limón • Jugo de naranja

Tabla de Anexo 1.17 Grupo de ingredientes de Jugo.

Alimento	Ingredientes
Pescado	• Pargo • Dorado • Cojinúa • Tiburón • Sardina • Atún • Cherna • Bonito

Tabla Anexo 1.18 Grupo de ingredientes de Pescado.

Alimento	Ingredientes

Marisco	• Camarón • Langosta • Pulpo • Calamar • Cobo • Ostión • Ostra

Tabla Anexo 1.19 Grupo de ingredientes de Marisco.

Alimento	Ingredientes
Salsa	• Salsa de tomate • Salsa de soya • Salsa china • Salsa de picante • Mayonesa

Tabla Anexo 1.20 Grupo de ingredientes de Salsa.

Alimento	*Ingredientes*
Pastas	• Espaguetis • Codito • **Macarrones**

Tabla Anexo 1.21 Grupo de ingredientes de Pastas.

Alimento	Ingredientes
Queso	• Queso amarillo • Queso blanco • Queso fundido

Tabla Anexo 1.22 Grupo de ingredientes de Queso.

Alimento	Ingredientes
Vegetal	• Habichuela • Remolacha • Zanahoria • Tomate • Lechuga • Col • Pepino • Acelga • Pimiento • Espinaca • Berenjena

Tabla Anexo 1.23 Grupo de ingredientes de Vegetal.

Alimento	Ingredientes

Vianda	• Papa • Yuca • Malanga • Calabaza • Plátano • Boniato • Guapeen • Ñame • Plátano burro

Tabla Anexo 1.24 Grupo de ingredientes de Vianda.

Alimento	Ingredientes
Condimento	• Pimiento rojo • Pimiento verde • Ajo • Cebolla • Comino • Orégano • Ají • Sal • Aceite • Azúcar • Pimienta • Puré de tomate • Picante • Canela • Chile • Sazonador chino • Cilantro • Cebollino • Mostaza • Laurel • Menta • Anís • Vinagre • Vino seco • Vino blanco • Vino tinto • Sofrito criollo • Limón • Naranja agria • Mantequilla

	• Salsa

Tabla Anexo 1.25 Grupo de ingredientes de Condimento.

Otros Ingredientes
• Huevo
• Agua
• Pan

Tabla Anexo 1.26 Grupo de Otros Ingredientes