ESCUELA SUPERIOR POLITECNICA DEL LITORAL
FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACION

TESIS DE GRADO

"Un Marco de Trabajo para el Desarrollo de Aplicaciones Web con Comportamiento Autónomo Inteligente"

Previa a la Obtención del Título de

Ingeniero en Computación

Especialización Sistemas de Información

Presentada por :

Otto Xavier Cordero Sánchez

GUAYAQUIL, 2003

ESCUELA SUPERIOR POLITECNICA DEL LITORAL

FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACION

TESIS DE GRADO

"Un Marco de Trabajo para el Desarrollo de Aplicaciones Web con Comportamiento Autónomo Inteligente"

Previa a la obtención del titulo de Ingeniero en Computación especialización Sistemas de Información

PRESENTADA POR:

Otto Xavier Cordero Sánchez

Guayaquil

2003

AGRADECIMIENTO.

Agradezco a mi madre por su esfuerzo y entrega hacia mí. Además a Monika quien siempre ha sido apoyo y equilibrio.

Otto X. Cordero S.

DEDICATORIA.

Este trabajo esta dedicado a mi madre.

Otto X. Cordero S.

TRIBUNAL DE GRADO.

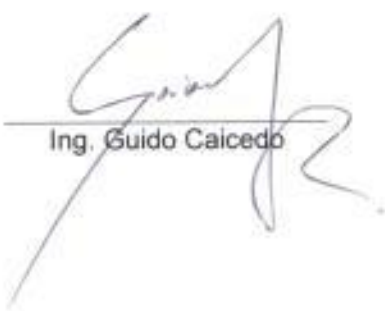
PRESIDENTE

Ing. Hernán Gutiérrez

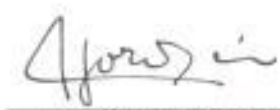
DIRECTOR DE TESIS

Dr. Enrique Peláez

MIEMBROS PRINCIPALES



Ing. Guido Caicedo



Ing. Carlos Jordán

DECLARACION EXPRESA.

"La responsabilidad por los hechos, ideas y doctrinas expuestas en esta tesis, nos corresponden exclusivamente, y el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Otto Cordero Sanchez

RESUMEN.

Este trabajo presenta un mecanismo para construir aplicaciones Web sensitivas a la información existente en el contexto. Es decir, sistemas capaces de reconfigurar su comportamiento en función del escenario en que se encuentren.

Se empieza analizando las características actuales del software basado en el web y se remarca la importancia del comportamiento autónomo. Posteriormente, se presentan las bases teóricas sobre computación sensitiva al contexto, conjuntos difusos y mapas cognitivos, que servirán para sustentar la solución propuesta.

El diseño de la plataforma se basa en la combinación de patrones de diseño de software con mapas cognitivos difusos, una herramienta para representar el conocimiento y realizar inferencias. Por tanto, se presenta un análisis de las tecnologías web actuales basadas en patrones y se selecciona una para la implementación de la plataforma. A continuación se detalla el diseño y la implementación del marco de trabajo y se analiza su eficiencia. Finalmente, se presenta un prototipo que muestra la flexibilidad alcanzada a través de la solución propuesta.

INDICE GENERAL.

INDICE DE FIGURAS

INTRODUCCION

Esta tesis presenta una alternativa para el desarrollo de aplicaciones web con cierto nivel de autonomía, que les permita reconfigurarse de acuerdo al contexto en que funcionen.

El objetivo principal de esta tesis es proponer las bases que permitan el desarrollo de aplicaciones web inteligentes, utilizando tecnologías de especificación abierta y bajo costo. Para lograr tal efecto se ha construido un marco de trabajo para aplicaciones sensitivas al contexto en el web. El eje principal de la solución se basa en dos principios: La subdivisión del software en pequeños componentes independientes y el uso de herramientas para la administración del conocimiento que permitan vincular conceptos de la lógica del negocio con los componentes antes mencionados, permitiéndole a los propietarios del sistema modelar su comportamiento a través de herramientas visuales.

Se han utilizado herramientas del proyecto Jakarta de la Fundación Apache, que soportan el uso de patrones de diseño y son de bajo costo, por mantenerse bajo un esquema de licencia libre. Además se utilizan otras tecnologías basadas en Java y software desarrollado por la ESPOL, en el Centro de Tecnologías de información.

CAPITULO 1

1. PERSPECTIVA ACTUAL

Con la popularización del Internet y el progreso de las tecnologías relacionadas, el desarrollo de software se ha ido enfocando en el ambiente Web, en donde los usuarios interactúan a través de herramientas (ej.: Internet Explorer) que les permiten visualizar múltiples documentos que combinan la lógica del negocio, los datos y la presentación visual. Sistemas de este tipo hoy en día se pueden encontrar en áreas tan complejas como la banca (www.bancodeguayaquil.com), las transacciones B2B entre empresas (www.covisant.com) o las ventas masivas a consumidores finales (www.amazon.com). En todos estos casos el volumen de la información así como la complejidad de los requerimientos y perfiles de usuarios requieren que estos sistemas muestren comportamiento especializado para cada escenario y actor especifico.

La orientación a objetos es uno de los paradigmas más relevantes en el desarrollo sobre el Web y los lenguajes más populares actualmente la soportan (Java y últimamente los lenguajes de la plataforma .Net). Uno de los principales resultados de la aplicación de esta metodología en el Web es la subdivisión de los sistemas en pequeños componentes y el énfasis en la arquitectura bajo la cual estos se integran.

Esta tesis se basa en la premisa de que las arquitecturas que se aplican en los sistemas basados en el Web establecen una lógica de interconexión de componentes estática y que tal hecho limita la capacidad de personalización del comportamiento de la aplicación. Como respuesta a esto se presenta una alternativa de ensamblaje del sistema en tiempo de ejecución. En esta línea, los diseñadores del sistema no establecen ya una arquitectura de componentes sino más bien un modelo causal bajo el cual estos se activan en función escenarios.

Para lograr tal efecto se propone la combinación de los Mapas Cognitivos Difusos, una herramienta para la administración del conocimiento basado en redes causales, y el patrón MVC, un esquema de subdivisión de la aplicación Web. Como se discutirá en el capitulo IV, este último ha recobrado popularidad y hoy en día existen marcos de trabajo con los cuales se puede contar para el desarrollo de sistemas bajo este modelo. Estos marcos de trabajo además se integran con repositorios de servicios

y componentes que cubren muchas de las funcionalidades comunes de una aplicación Web (véase el subproyecto Fulcrum de la fundación Apache, http://jakarta.apache.org).

Por tanto, actualmente están sentadas las bases para construir un marco de trabajo que permita construir aplicaciones Web más flexibles, capaces de reconfigurarse en tiempo de ejecución para satisfacer necesidades específicas del escenario en el cual se encuentran. La tesis que aquí se presenta propone un mecanismo para lograr este objetivo.

1.1 Importancia Del Comportamiento Autónomo

La posibilidad de tener un sistema que se reconfigure en tiempo de ejecución de acuerdo a los escenarios en los que se encuentra, tiene implícita la necesidad de que exista cierto nivel de autonomía que le permita escoger cual configuración aplicar de acuerdo al contexto. La autonomía es entonces un elemento esencial en las aplicaciones con comportamiento inteligente y su estudio y desarrollo probablemente tenga uno de los impactos más significativos en los sistemas del futuro. Actualmente una de las iniciativas más populares es la conocida como *Autonomic Computing* (www.research.ibm.com/autonomic) de IBM. El objetivo de ese proyecto es construir maquinas capaces de ejecutarse a si mismas, ajustarse a circunstancias variables y preparar sus recursos para manejar de forma más eficiente las cargas de trabajo

que ponemos sobre ellas. Estos sistemas autónomos deben anticipar las necesidades y permitir a los usuarios enfocarse en lo que quieren lograr, más que en como hacer que los computadores hagan lo que ellos desean.

Los sistemas autónomos tienen muchas implicaciones desafiantes para la tecnología actual, por ejemplo la necesidad de que estos se "conozcan" a si mismos y a sus partes. Sin embargo, hoy en día se están desarrollando pequeños avances especialmente en área del manejo de datos y flujos de trabajo. Un requerimiento esencial de este tipo de sistemas es la capacidad de utilizar información en el contexto, tal como ubicación, emociones y hechos pasados, para autoconfigurarse. En este aspecto lo que se conoce como computación sensitiva al contexto es una de las áreas que ofrece mayores oportunidades para encontrar mecanismos que nos permitan ver sistemas autónomos en un futuro cercano. En laboratorios reconocidos como el MIT Media Lab (www.media.mit.edu/research) se estudia este tema y se desarrollan artefactos que reaccionan a la información contextual.

Uno de los objetivos de esta tesis es llevar la computación sensitiva al contexto al campo de los sistemas basados en el Web. Para lograr esto los modelos causales serán la herramienta principal, dado que en

base a ellos se pueden realizar inferencias a partir de la información presente en el contexto y obtener conclusiones acerca del comportamiento a seguir. Con esta herramienta a mano, esta tesis presenta un marco de trabajo que puede servir de punto de partida para lograr aplicaciones Web autónomas, capaces de "tomar decisiones" acerca de su propio funcionamiento en virtud de datos del contexto tales como perfiles de usuarios, estado del negocio, carga de la red, etc.

El tema de la computación sensitiva al contexto se trata en detalle durante el Capitulo II en base a sus fundamentos teóricos. Además, se muestra como esta puede ser aplicada en el Internet y cuales serían los principales puntos a considerar en una aplicación Web sensitiva al contexto. Finalmente, se mencionan estrategias para el manejo de la información del contexto en el Web.

El desarrollo de la base teórica continúa en el Capitulo III con la descripción de la teoría de conjuntos difusos, los Mapas Cognitivos Difusos y su aplicación en el control de sistemas. En este capítulo también se muestra la potencial aplicación de los Mapas Cognitivos Difusos en sistemas dinámicos de e-Business como herramientas para la administración del conocimiento y modelos de comportamiento autónomo.

En el capítulo IV se analizan las tecnologías Web y los patrones arquitectónicos, especialmente el patrón MVC y las tecnologías que lo soportan por ser este el de mayor acogida. Además, se sustenta la elección de la tecnología usada en el prototipo. El capítulo V se muestra como se pueden fusionar el patrón MVC y los mapas cognitivos para ensamblar la aplicación Web de forma dinámica de acuerdo a las reglas contenidas en los mapas cognitivos.

En el capítulo VI se muestra una aplicación prototipo donde se prueban las ideas propuestas en una aplicación Web sociable. El sistema responde con aparentes emociones a diferentes estados internos. Este prototipo esta inspirado en trabajos previos en sistemas sociables y es una implementación a pequeña escala que muestra el potencial de la plataforma. Finalmente, el Capítulo VII presenta las conclusiones y recomendaciones para trabajos futuros.

CAPITULO 2

2. COMPUTACIÓN SENSITIVA AL CONTEXTO EN EL WEB

2.1 Introducción a la Computación Sensitiva al Contexto

Las aplicaciones sensitivas al contexto son aquellas que cambian su comportamiento de acuerdo a la información del contexto (2). En una aplicación web la información del contexto puede ser utilizada para adaptar interfaces, contenidos y transacciones.

Hoy en día los sistemas computacionales están tomando un giro importante y los diseñadores están considerando la sensitividad al contexto como una necesidad importante. El hardware y software han sido conceptualizados como sistemas de entrada-salida: Sistemas que toman datos ingresados por un usuario y utilizan esa entrada por si sola para producir la respuesta. Esta vista es hoy muy restrictiva. Los dispositivos inteligentes del futuro mediato operarán con datos que no

han sido otorgados explícitamente, sino que más bien han sido observados u obtenidos por ellos mismos.

El contexto se define como la situación en el entorno social o físico en el que se encuentra un objeto. En el caso de los sistemas computarizados este se define como "cualquier información que puede ser utilizada para caracterizar la situación de una entidad, donde una entidad puede ser una persona, lugar u objeto físico o computacional" (2). En base a esto se ha definido la computación sensitiva al contexto como "El uso del contexto para proveer información relevante a las tareas y/o servicios al usuario, donde sea que estos estén" (2). De aquí se concluye que existen tres comportamientos significativos que una aplicación sensitiva al contexto puede mostrar:

1. La presentación de información y servicios al usuario
2. La ejecución automática de servicios
3. El análisis del contexto para la obtención de información y su posterior uso.

Las aplicaciones sensitivas al contexto son especialmente relevantes en el área de los sistemas omnipresentes y ubicuos (pervasive & ubiquitous). Los sistemas omnipresentes están, tal como su nombre lo indica, ubicados en todos o casi todos los escenarios en los cuales

participa el usuario, pueden estar embebidos en electrodomésticos, teléfonos, autos, computadores, semáforos, puertas, ascensores, etc. Estos sistemas además pueden estar organizados en federaciones en las cuales existe cierto grado de comunicación y cooperación. Los sistemas ubicuos son invisibles al usuario, es decir, este se beneficia de su uso sin percibir su presencia. Por ejemplo, ciertos sistemas de seguridad utilizan tarjetas electrónicas que contienen chips. Estos chips son invisibles al usuario y éste, aunque percibe los beneficios de la tecnología, no esta conciente de su presencia.

Esta tesis intenta llevar las aplicaciones Web al campo de los sistemas sensitivos al contexto, considerando la posibilidad de que las aplicaciones web actuales no obtengan el máximo provecho de la información contextual, y de que los mecanismos actuales no sean suficientes para utilizar efectivamente y en línea la inmensa cantidad de datos a las que tienen acceso los programas situados en la red. En la siguiente sección se presenta la computación sensitiva al contexto en el escenario del Web.

2.2 Sensitividad al Contexto en Aplicaciones Web

Los sitios y aplicaciones web actuales son diseñados como software dinámico, donde la presentación, el contenido y las reglas del negocio son combinados bajo demanda. Sin embargo, esta clase de aplicación

utiliza una lógica estática para combinar sus elementos, sin considerar variables del ambiente del negocio, preferencias y hábitos de usuario, información de perfiles, etc. Como resultado, la aplicación posee vínculos dinámicos entre sus componentes, pero vínculos estáticos con constituyentes externos como clientes u otros negocios en línea (4).

En la medida en que los ambientes de negocio se tornan más competitivos, la especialización de la interacción con el usuario y la automatización de las decisiones se convierten en metas importantes. Para lograr esto las aplicaciones web deben usar la información del contexto. Esta información esta normalmente disponible en las aplicaciones web que recogen grandes cantidades de datos a partir de formularios Web de registro, transacciones, hábitos de búsqueda y manejo de información, etc.

Las principales fuentes de información del contexto en aplicaciones web son:

- Datos de la sesión del usuario almacenados en cookies
- Historia de las transacciones y perfil del usuario, normalmente almacenada en bases de datos.
- Información del ambiente como por ejemplo el tráfico de la red.
- Datos del negocio (información del inventario, ofertas, etc.)

Las aplicaciones Web pueden ser representadas de manera teórica en un espacio multidimensional, donde las dimensiones representan los parámetros de comportamiento del sistema. Como se observa en la Figura 2.1, una aplicación estática es representada por un punto fijo en el espacio, mientras que una aplicación sensitiva al contexto cambia constantemente su estado en el tiempo según los cambios en el ambiente externo (21)

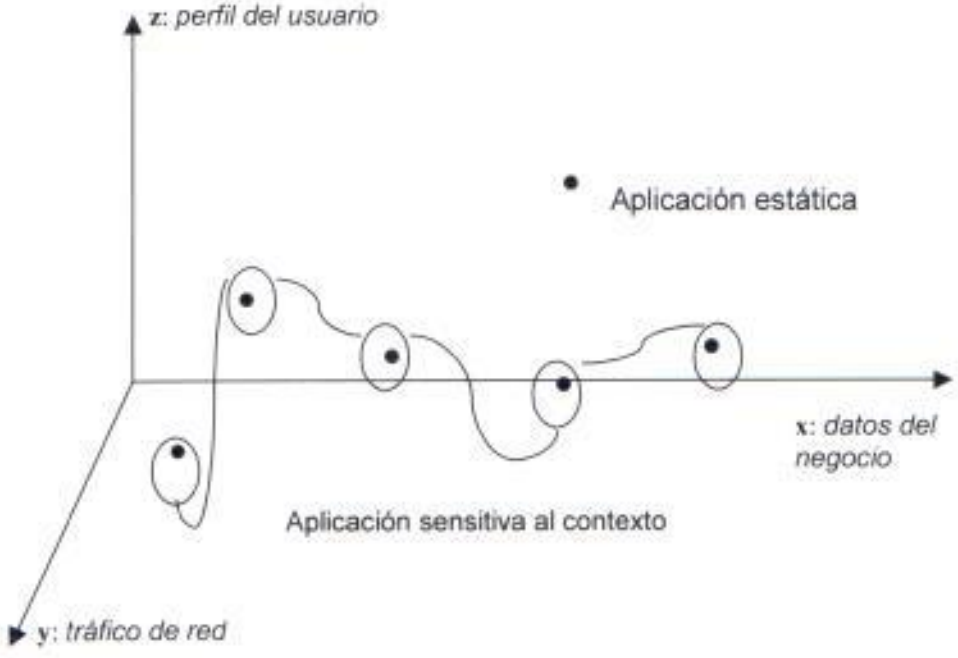


Figura 2.1. Las aplicaciones web en un espacio multidimensional.

De forma general se puede decir que una aplicación web sensitiva al contexto debe basarse en los siguientes criterios de diseño:

- Independencia en los elementos que componen la aplicación web.
- Capacidad de combinar los elementos de forma dinámica.
- Una lógica capaz de vincular los datos provenientes del contexto con el estado de la aplicación Web.

Como se observará en los capítulos subsiguientes, actualmente existen marcos de trabajo que permiten independizar los elementos de los cuales se componen las aplicaciones Web, y combinarlos de forma dinámica para construir la aplicación en tiempo real. Adicionalmente, se presenta el uso de los mapas cognitivos difusos como una herramienta capaz de proveer la lógica que conduzca el comportamiento sensitivo al contexto y vincule este con el conocimiento dentro de las organizaciones.

2.3 Estrategias para el Manejo de la Información del Contexto

Existen dos tareas principales en el manejo de la información del contexto: i) su adquisición a partir de las fuentes y ii) su representación. Como se mencionó en la sección anterior existen cuatro fuentes de información del contexto en aplicaciones web, para cada una de éstas la tarea de adquisición tiene diferentes características. En cualquiera de estos casos, el primer punto consiste

en seleccionar qué información es relevante para adecuar el comportamiento de la aplicación Web. Ejemplos de información relevante son datos demográficos del usuario, frecuencia de visitas, tipo de software que utiliza, etc.

Algunos de los datos que constituyen la información relevante del contexto se pueden encontrar de forma explicita en las fuentes y otros quizás necesiten ser procesados. La obtención de estos datos debe estar encapsulada en un modulo que se encargue de traducir la información de las fuentes a los elementos de la representación escogida para el contexto.

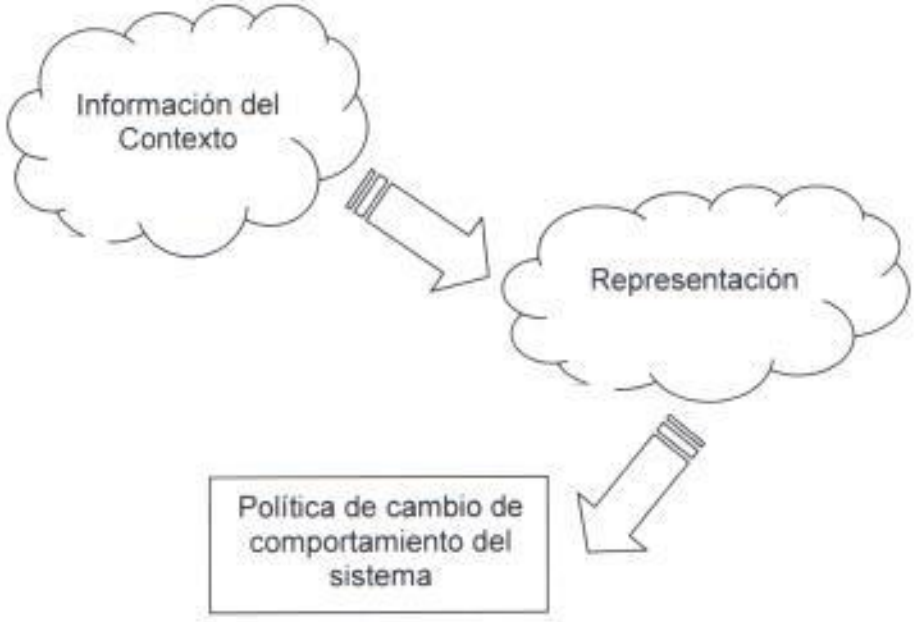


Figura 2.2. El manejo de la información del contexto.

Esto nos lleva al siguiente punto importante que es la representación de la información del contexto. De acuerdo con Zaslavsky (21) la información del contexto cae en un espacio N-dimensional donde N es el numero de aspectos del contexto:

$$C = \{c_1, c_2, c_3, ...c_N\}, c_i \in C_{ambiente} \vee C_{usuario} \vee C_{negocio} \quad (2.1)$$

En este caso la información del contexto que se obtiene de las fuentes antes mencionadas puede caer dentro de tres aspectos, el contexto del ambiente (Ej.: trafico de la red), el contexto del usuario (perfil del usuario, historial de transacciones y datos de la sesión) y el contexto del negocio (información de los inventarios, ofertas, índices internos, etc.).

Considerando el espacio *C* de información del contexto y el espacio *A* de posibles acciones que cambien el comportamiento de la aplicación web, el punto clave para producir una aplicación sensitiva al contexto consiste en encontrar una política π^*: $C \rightarrow A$ que permita seleccionar las acciones que modifiquen de forma optima el comportamiento de la aplicación web según el contexto, es decir que, $\pi^*(c_1, c_2, c_3, \ldots, c_N) = (a_1, a_2, a_3, \ldots, a_M)$, donde $a_i \in A$.

El problema consiste en definir que es un "comportamiento optimo" para una aplicación web, pues no existe una función de evaluación cuantitativa para esto. El rendimiento de la aplicación merece una apreciación subjetiva por parte del usuario y de los promotores del negocio detrás de la aplicación.

En los siguientes capítulos se describe una estrategia basada en administración del conocimiento para permitir a los expertos dentro de la organización desarrollar modelos de comportamiento del negocio, de forma que estos aproximen lo que se estima es el comportamiento optimo de la aplicación.

CAPITULO 3

3. MODELOS COGNITIVOS Y CONJUNTOS DIFUSOS PARA LA TOMA DE DECISIONES

3.1 Conjuntos Difusos: Breve Introducción

En la teoría de conjuntos tradicionales (no difusos) cada conjunto esta definido por una función que asigna un valor de 1 o 0 a cada elemento en el conjunto universo, con el objeto de discriminar entre elementos y no-elementos del conjunto en cuestión. Esta función puede ser generalizada de forma que el valor asignado a los elementos del conjunto universo este dentro de un rango que indique el grado de membresía del elemento en un conjunto dado. Tal función se conoce como *función de membresía*, y el conjunto que esta define es un *conjunto difuso* (11).

El rango más común para definir la función de membresía es el intervalo continuo [0,1], donde 0 expresa no membresía absoluta en el

conjunto y 1 expresa membresía absoluta en el conjunto. Los valores entre 0 y uno representan grados parciales de membresía.

Para denotar las funciones de membresía usaremos la notación $\mu_A : X \rightarrow [0,1]$ Donde X es el universo de elementos, A es el conjunto difuso y μ_A es la función de membresía del conjunto A (11).

La Figura 3.1 ilustra la comparación entre los conjuntos tradicionales y los conjuntos difusos:

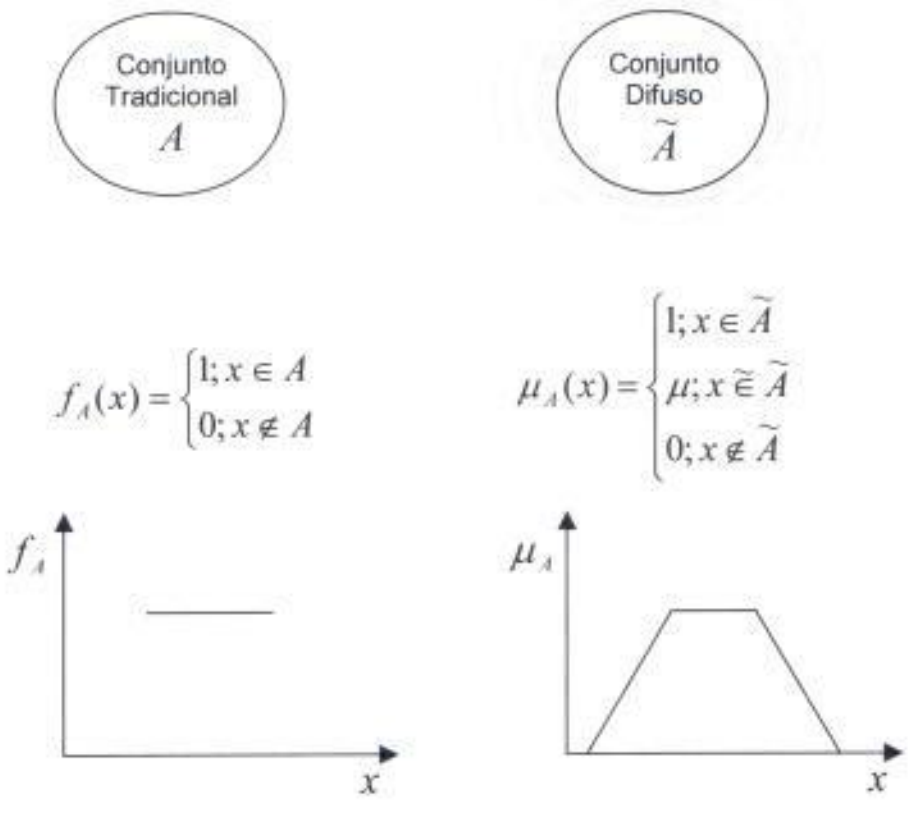


Figura 3.1. Comparación entre conjuntos tradicionales y difusos.

Donde μ es el grado de membresía de x en $\tilde{A}$ y $\tilde{\in}$ representa la pertenencia parcial de x en $\tilde{A}$. Finalmente se muestran las funciones de membresía en el plano cartesiano, donde la función de membresía del conjunto tradicional describe una recta, mientras que la función de membresía del conjunto difuso tiene forma convexa. Según el contexto, la función de membresía puede ser de tipo triangular, en campana o trapezoidal como en la Figura 3.1.

La capacidad de los conjuntos difusos de expresar transiciones graduales de membresía a no-membresía y viceversa, nos provee no solo una representación poderosa de la incertidumbre en las medidas, sino también una representación de los conceptos vagos que manejamos en el lenguaje natural. Por ejemplo, podríamos describir el clima con un término tal como *soleado* o *nublado*, en vez de expresar el porcentaje exacto de nubes en el cielo. Claramente esto es más útil pues raramente contamos con datos tan específicos. Así podríamos manejar términos tales como conjuntos difusos, donde las funciones de membresía se describen sobre el dominio del nivel porcentual de nubes. Entonces, por ejemplo, nuestro conjunto difuso describiendo el concepto *soleado* asignará un nivel de membresía 1 al nivel porcentual de nubes 0%. , 0.8 a un 20%, .4 a un 50% y 0 al 75%.

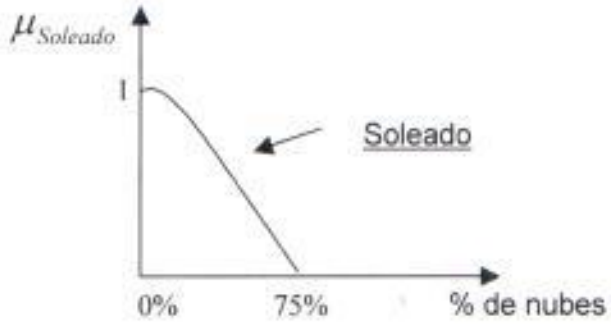


Figura 3.2. Ejemplo de un conjunto difuso

De esta forma, los términos *soleado* y *nublado* seria lo que se conoce como variables lingüísticas (11). Estas son variables cuyos estados son conjuntos difusos que representan conceptos lingüísticos, como *muy pequeño, pequeño, mediano*, etc. Cada variable lingüística tiene una variable base, un rango de números reales sobre los cuales se construye el conjunto difuso. La variable base es una variable en el sentido clásico, al igual que cualquier medida física como temperatura, presión, velocidad, voltaje, etc. Cualquier variable lingüística puede ser definida de forma matemática por una quíntupla (v, T, X, g, m) en la cual v es el nombre de la variable, T es el conjunto de términos lingüísticos de v que se refieren a la variable base cuyos valores están en el rango del conjunto X, g es una regla sintáctica para generar términos lingüísticos, y m es una regla semántica que asigna a cada termino lingüístico $t \in T$ su significado, el cual es un conjunto difuso en X. En la siguiente sección usaremos este concepto para entender

los mapas cognitivos difusos y su aplicación en el modelamiento de las relaciones causales.

3.2 Mapas Cognitivos Difusos

Los Mapas Cognitivos fueron introducidos por Axelrod y han sido usados para representar conocimiento en ciencias políticas y sociales (1). Los Mapas Cognitivos proveen una representación distribuida de las relaciones causa-efecto de un sistema, que puede ser usada para realizar inferencias. Además, poseen una naturaleza gráfica que permite la adquisición directa del conocimiento y su evaluación.

En un Mapa Cognitivo, conceptos tales como "Estrategia Agresiva", "Servicio Personalizado", etc. son representados como nodos en un grafo dirigido. Las flechas dirigidas representan relaciones causales entre conceptos. Las relaciones marcadas como positivas indican que un incremento en la presencia de un concepto lleva a un incremento en el otro, y las que han sido marcadas como negativas indican que un incremento en la presencia de uno hace que el otro disminuya (14). Los Mapas Cognitivos Difusos (FCM, por sus siglas en inglés) extienden el concepto de los Mapas Cognitivos, permitiendo representar pesos causales con cuantificadores lingüísticos difusos definidos en el intervalo [-1,1] (12).

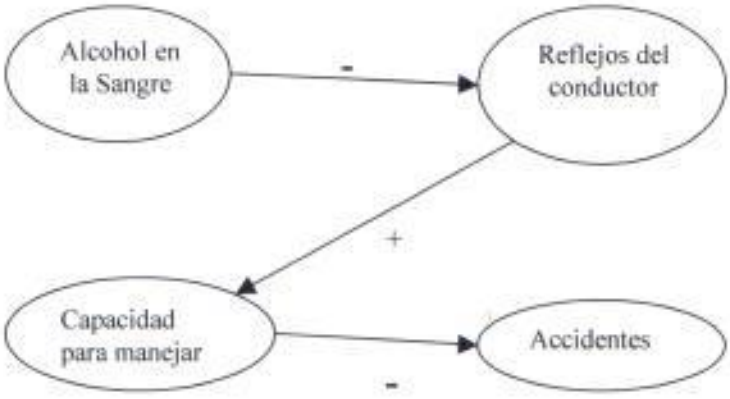


Figura 3.3. Mapa cognitivo sobre las causas de los accidentes

En la Figura 3.3 se muestra un mapa cognitivo hipotético sobre las causas de los accidentes de transito. Como se observa, las conexiones han sido marcadas con signos + ó - para indicar relaciones causales positivas o negativas respectivamente. En un mapa cognitivo difuso, adicionalmente estas conexiones tienen "pesos", es decir cuantificadores que expresan el nivel de certeza de la relación entre los conceptos. Estos cuantificadores pueden ser variables lingüísticas o números reales. En la Figura 3.4 se muestra el mismo mapa cognitivo, pero ahora con pesos en sus conexiones.

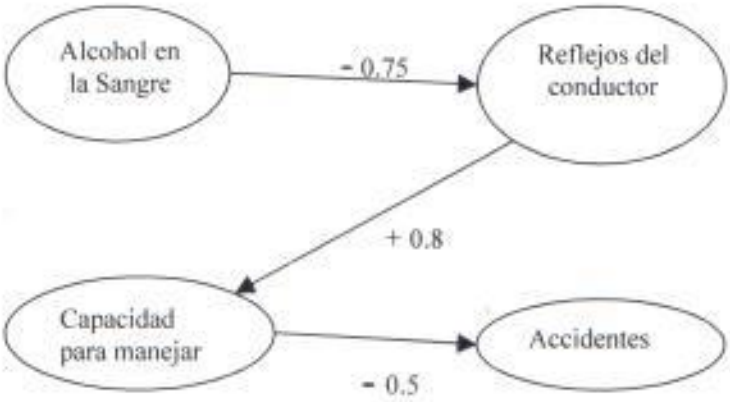


Figura 3.4. Mapa cognitivo difuso sobre las causas de los accidentes

Los mapas cognitivos, dado que son esencialmente grafos, pueden ser representados con matrices de conexiones, esto es, matrices cuadradas en las cuales la fila y la columna i representan un nodo del grafo. Un elemento en la celda i, j representa la conexión entre el concepto i y el concepto j. En un grafo cualquiera de estos elementos serán 1 ó 0 según exista o no la conexión entre los nodos. En un mapa cognitivo difuso, los elementos de la matriz son números reales en el intervalo [0,1], estos números expresan el peso de la relación entre los conceptos. La Figura 3.5 muestra una matriz de conexiones para el mapa mostrado anteriormente. Como se observa en esta figura, los elementos de la diagonal de la matriz siempre son 1.

$$\begin{bmatrix} 1 & -0.75 & 0 & 0 \\ 0 & 1 & 0.8 & 0 \\ 0 & 0 & 1 & -0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 3.5. Ejemplo de una matriz de conexiones.

Para examinar el comportamiento dinámico del FCM, el valor de activación de un concepto en un instante de tiempo es representado por un vector de estado, y las relaciones entre conceptos por una matriz de conexiones como la que se muestro en la Figura 3.5. Un concepto es "encendido" al hacer 1 su elemento en el vector de estado. Entonces, el vector es iterativamente multiplicado por la matriz de conexiones usando la multiplicación estándar de matrices. La iteración se detiene cuando se alcanza un vector de estados del cual no se puede inferir nada nuevo, es decir, cuando se estabiliza el proceso dinámico de inferencia (6).

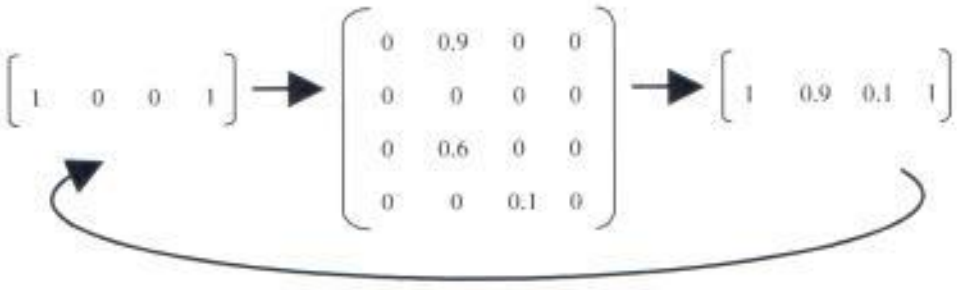


Figura 3.6. El proceso de inferencia con los mapas cognitivos difusos

La Figura 3.6 muestra una representación simbólica del proceso de inferencia. Al inicio tenemos un vector cuyos elementos representan el estado de cada uno de los conceptos al iniciar la inferencia (activado = 1, desactivado = 0), este vector es luego multiplicado por la matriz de conexiones. El resultado de esta multiplicación es otro vector que será nuevamente multiplicado por la misma matriz y así sucesivamente hasta que el vector i + 1 sea igual a cualquier otro vector obtenido anteriormente durante el proceso. Si el vector i + 1 es igual al vector i, el sistema se ha estabilizado en un estado final y nada nuevo puede inferirse. Si el vector i + 1 es igual a cualquier otro vector obtenido anteriormente, esto implica que se ha detectado un ciclo y que no se producirán nuevos vectores por lo que nada mas será inferido.

3.3 Hacia un "e-Business Intelligence" Dinámico

Hoy en día, la automatización de las reglas del negocio así como de las decisiones tomadas en base a estas reglas, es un objetivo importante en los escenarios de cualquier negocio, especialmente en aquellos vinculados con el comercio electrónico. Sin embargo, el ambiente dinámico que los rodea dificulta encontrar un formato para expresar este conocimiento (4).

Las estrategias de "Business Intelligence" (BI) están principalmente enfocadas en la generación de información con valor, usando análisis

inteligente de datos y comunicación multi-direccional de la información, para dar poder a los tomadores de decisiones con nuevo conocimiento (13).

El rol del BI es la transformación de los datos en información valiosa y accesible. A mayor el valor y accesibilidad de la información, mayor es el potencial de conocimiento usable de los usuarios habilitados con BI. Aun cuando no se puede afirmar que las tecnologías por si mismas sean capaces de incrementar el conocimiento o la información disponible para los usuarios, estas al menos crean los canales de información que, bajo un enfoque organizacional orientado ha explotarlas, pueden servir para obtener información que mejore la calidad de las decisiones

Como ejemplo se muestra el caso de la implementación de un sistema para la planeación de la producción en Novartis (13). Novartis es el resultado de la fusión de dos de las mayores compañías farmacéuticas, Ciba-Geigy y Sandoz. Uno de los principales retos de la división de agro-químicos es que la planeación sea hecha con cinco años de anticipación. Esto es una tarea muy complicada, que implica la integración y el análisis de gran cantidad de información externa, como por ejemplo pronósticos a largo plazo de ventas y mercado. Adicionalmente existen ciclos que deben ser tomados en cuenta. El

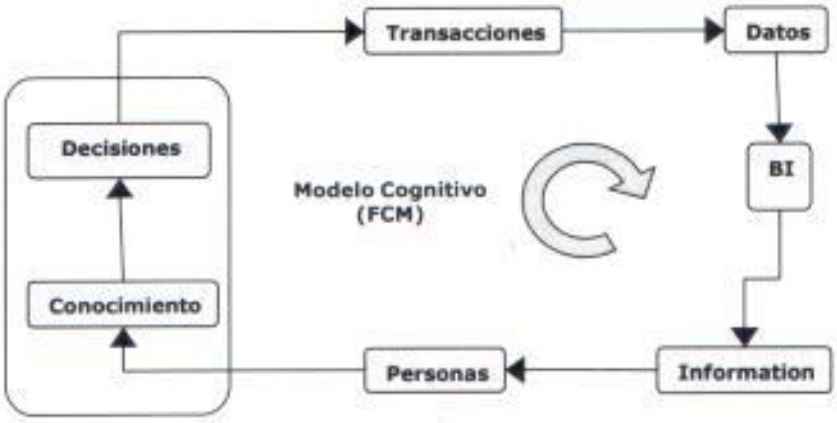
manejo de la producción y proceso de distribución es también crítico y complejo pues implica múltiples substancias y materiales que involucran numerosos proveedores. El objetivo de la empresa era obtener control sobre todos los datos relevantes. La compañía decidió estandarizar sus módulos de finanzas, materiales, producción, etc. en un solo sistema de BI basado en un data-warehouse. Uno de los principales requerimientos de la nueva solución es que debía ser usable para todos dentro de la organización sin importar posición o nivel técnico (basándose en el principio de que a mayor accesibilidad de la información y mayor intercambio de información entre departamentos y funciones, mejores resultados). Luego de la implementación del sistema, la empresa es capaz de producir reportes que antes tomaban semanas en días y ha logrado tomar decisiones que aceleran la planeación de forma critica. Otros casos que pueden ser revisados en (13) y que ilustran este principio son, Ingram Micro (mayorista de productos tecnológicos), Penske Logistics (transporte y carga), Zurich U.S (seguros comerciales), BOC Gases (gas industrial), Eli Lily (farmacéutica), etc.

El valor de las decisiones se incrementa al disminuir el tiempo que toma alcanzarlas o implementarlas. Adicionalmente, de acuerdo a Business Objects, la compañía líder en soluciones de BI, el 80% del tiempo del ciclo de decisiones se utiliza en recolectar la información,

mientras que el 20% restante es tiempo real de toma de decisiones (13). Por tanto, el principio de incrementar la accesibilidad (tiempo de acceso y volumen) de la información tiene una base lógica, que dentro de una cultura organizacional y técnica adecuada, puede dar resultados tales como los que se ilustran en los casos antes mencionados.

Como se ha indicado antes, las estrategias de BI en combinación con una cultura organizacional adecuada facilitan la generación de conocimiento a partir de los datos, al convertir éstos en información significativa y de acceso eficiente para los usuarios y las organizaciones. Este conocimiento es, sin embargo, único y personal, ya que es el resultado de la percepción, sentimientos, y antecedentes culturales distintivos de cada persona.

Los mapas cognitivos son capaces de manejar este conocimiento, permitiendo la adquisición, combinación y representación de éste de una forma sencilla. Si este modelo es incluido en el ciclo de datos (Figura 3.7), podemos obtener una relación sinérgica entre las estrategias de BI y los modelos cognitivos, al proveer un acercamiento no solo capaz de convertir los datos en información sino también el conocimiento en acciones de negocio (3).



Fuente: (4)

Figura 3.7. El rol de los mapas cognitivos en el ciclo de datos.

En escenarios de e-commerce, las decisiones asistidas por modelos cognitivos pueden ser fácilmente asociadas con reglas y acciones de negocio, dado que estas son manejadas por software. Esto permitiría la selección en tiempo real de acciones, en respuesta a entradas externas. Así podríamos transferir la inteligencia hacia constituyentes externos, permitiendo reacciones inteligentes de acuerdo a entradas y reglas de negocio específicas, codificadas en la representación cognitiva.

3.3.1 Adquisición del Conocimiento

La estructura de los FCM simplifica la adquisición del conocimiento, permitiendo a los expertos expresar su conocimiento en forma gráfica y en un marco de trabajo interactivo. Esta representación del conocimiento puede ser implementada en ambientes colaborativos, donde un grupo de expertos modela su conocimiento en los mapas sobre la base del consenso y la discusión.

Hay básicamente dos tareas en la construcción del FCM (14):

- La adquisición del conocimiento.
- El análisis del FCM una vez que ha sido construido.

Podemos dividir el conocimiento dentro de una organización en diferentes niveles de abstracción, desde conceptos corporativos de alto nivel hasta acciones operacionales de bajo nivel. En cada nivel, la adquisición del conocimiento se realiza en tres etapas: definición de conceptos, definición de conexiones y definición de los pesos en las conexiones entre conceptos (véase sección 3.2) (14).

Los FCM que representan diferentes niveles conceptuales en una organización deben estar vinculados para permitir fluir hacia abajo a los conceptos de alto nivel hasta controlar los procesos del negocio. Dos mapas están vinculados si estos comparten al menos un nodo-concepto. Luego, si el proceso de inferencia en uno o más de estos mapas activa uno de los conceptos compartidos, el flujo causal se transmite a los otros mapas, produciendo estados estables que combinan los conceptos de los mapas vinculados. En la Figura 3.8 se provee una ilustración de este fenómeno, donde tres mapas diferentes están vinculados bidireccionalmente. Este tipo de esquemas permiten conectar mapas que representan diferentes tópicos en un solo proceso de inferencia.

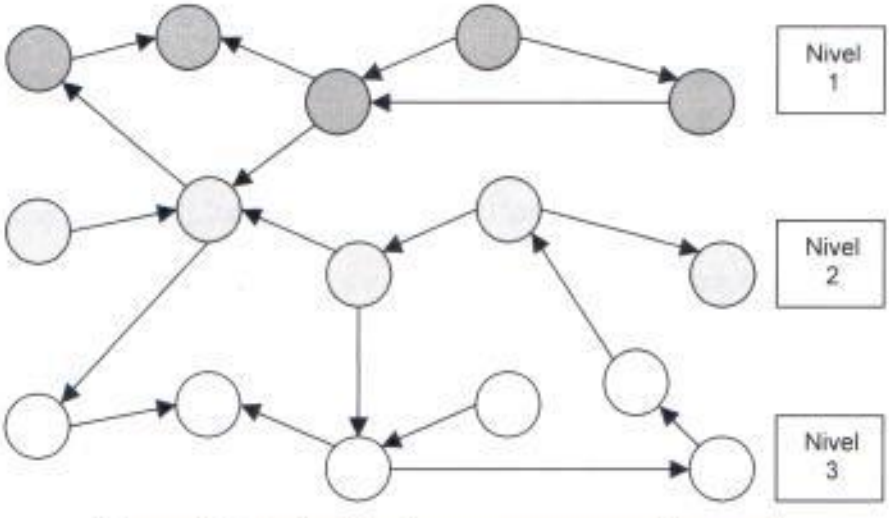


Figura 3.8. Ilustración de un esquema multi-capa de representación del conocimiento con mapas cognitivos

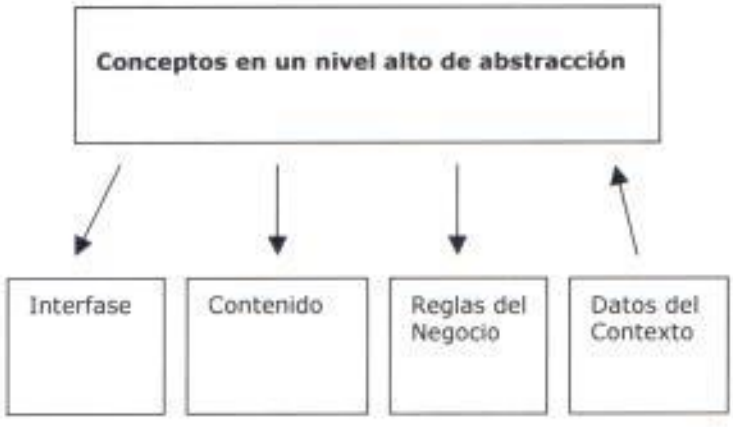
Una vez que el FCM ha sido construido, debe ser analizado para asegurar la estabilidad del sistema. El sistema es llamado estable si este converge a un estado controlado cuando cambios externos son introducidos a sus variables. En términos de los FCM, esto quiere decir que el proceso de inferencia no llega nunca a un vector de estados final. La detección de la instabilidad es importante pues es una indicación de la ausencia de relaciones criticas entre los conceptos (14). La inestabilidad es más probable que ocurra en FCMs que describen conocimiento de alto nivel, donde la expresión de conceptos y relaciones difíciles de medir objetivamente puede llevar a una sobrecarga de lazos cíclicos. En tales casos, los conceptos van a permanecer oscilando en un patrón de alto nivel que no producirá ninguna acción concreta. Tomemos por ejemplo el problema de los accidentes de transito, un mapa en un nivel de abstracción alto podría involucrar conceptos tales como "perdida de control bajo presión", "concentración débil", "problemas domésticos", etc. Todos estos conceptos se prestan para interpretaciones en las que pueden ser relacionados unos a otros entre si, provocando ciclos en el mapa. Mientras que si tenemos conceptos más concretos, es decir que pueden ser apoyados en datos o bases objetivas, es más fácil eliminar ambigüedades en las relaciones.

Por ejemplo, "Conduce durante horas pico", "Escucha música o conversa mientras maneja", etc. podrían ser causas indirectas de accidentes, pero no formarían parte de relaciones cíclicas entre causas y efectos.

3.3.2 Estructura del Conocimiento

El conocimiento en base al cual se modela el comportamiento de las aplicaciones web (Ej.: reglas de negocio) tiene una estructura multi-nivel, donde la capa más alta posee conexiones en el mapa cognitivo que involucran conceptos abstractos, tales como metas y decisiones corporativas. En el nivel mas bajo, el proceso de inferencia empieza y termina, cuando algunos conceptos son activados por entradas tomadas de la aplicación, activando las conexiones y ciertos conceptos de otros mapas. Finalmente, las conexiones llevan a nodos finales (nodos de los cuales no salen conexiones) en el mapa de más bajo nivel, estos nodos representan cambios en el estado de la aplicación. El FCM esta esperando por eventos para empezar el proceso de inferencia (Ej.: una petición de catalogo), entonces es capaz de llevar el flujo del conocimiento hasta los conceptos del negocio antes de seleccionar interfaces, contenidos y acciones. Véase la ilustración de la Figura 3.9, en donde los conceptos de más alto

nivel que controlan las reglas del negocio, la generación de contenidos e interfase, son activados por cambios en los datos del contexto, formando un ciclo en donde la información fluye hacia arriba y luego regresa para producir cambios de estado en la aplicación.



Fuente: (5)

Figura 3.9. La Estructura del Conocimiento

3.3.3 Inteligencia Basada en la Colaboración

Un factor importante en el BI es la colaboración entre los individuos. Las estrategias actuales se enfocan en dar poder a los tomadores de decisiones a través del compartimiento multidireccional de la información (13).

Los mapas cognitivos pueden ser interpretados como una representación distribuida del conocimiento, donde muchos

expertos, independientemente de su ubicación geográfica, pueden participar de la construcción del conocimiento y del proceso de toma de decisiones.

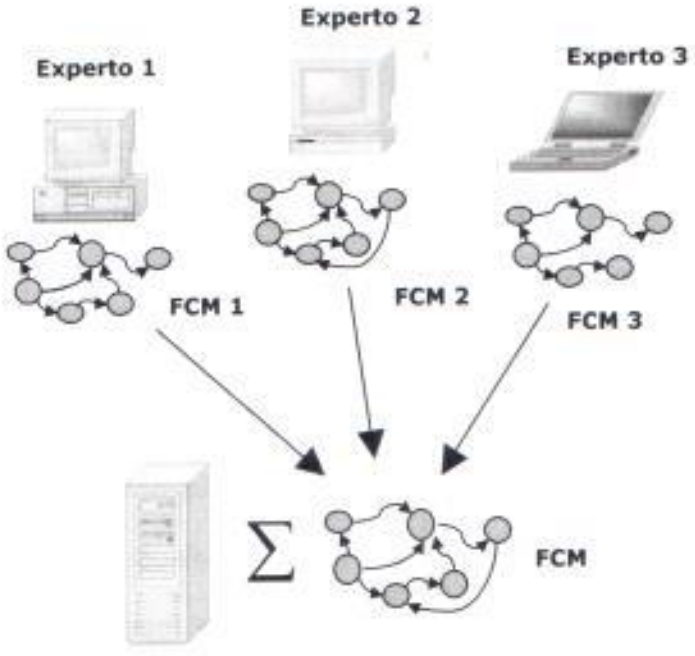


Figura 3.10. Colaboración en la construcción del conocimiento con FCMs

Debido a que los mapas cognitivos son básicamente grafos, es posible construir mapas de forma distribuida para luego combinarlos en una representación unificada, como se muestra en la Figura 3.10. Esto permite trabajar de forma colaborativa y simultanea al grupo de expertos. En una empresa puede haber

expertos trabajando en mapas que modelen el conocimiento relacionado a marketing, finanzas, operaciones, etc. de forma independiente, y luego estos mapas podrían ser combinados para obtener el modelo unificado del conocimiento de la empresa.

Los mapas pueden ser combinados de dos formas: a) conectando mapas a través de nodos comunes y b) sumando mapas. Dos mapas están conectados siempre que tengan un nodo en común, de esta forma el flujo causal se transmite de un mapa a otro a través de uno o más nodos comunes (Véase Figura 3.8). Otro aspecto interesante es la suma de mapas, esto es la superimposición de las matrices de conexiones de cada mapa, sumando los valores de las conexiones. Los mapas pueden ser ponderados según la credibilidad del experto, de forma que un mapa hecho por un experto con más crédito tenga mayor peso en el mapa final. Podemos combinar una matriz de conexiones cualquiera F_i con credibilidad w_i ($w = 1$ indica máxima credibilidad y $w = 0$ mínima credibilidad) simplemente al multiplicar $w*F_i$ y sumarla con otra matriz w_j*F_j. Incluso matrices con diferente número de nodos pueden ser combinados al introducir matrices aumentadas, o en otras palabras completando las filas y columnas de los nodos faltantes con 0s. De esta forma

podemos obtener matrices combinadas que representen la unión de uno o más mapas, y sobre las cuales se puede realizar inferencias que involucren todos los conceptos (12).

CAPITULO 4

4. TECNOLOGÍAS Y PLATAFORMAS WEB

4.1 Patrones Arquitectónicos

Actualmente, los proyectos basados en tecnologías web alcanzan grados de complejidad muy altos, incluyendo diversas tecnologías y tareas tales como el manejo y generación dinámica de múltiples documentos HTML, conexiones con bases de datos, manejo de seguridades y usuarios, administración de contenidos, ventas en línea, etc. Por esta razón, es muy conveniente utilizar plataformas que resuelven muchos de los problemas estándares y otorgan un modelo sobre el cual construir. Tales plataformas se presentan normalmente en forma de *capas* que encapsulan funcionalidades, como por ejemplo: conexión segura a bases de datos, manejo de plantillas de diseño, etc. Las plataformas sobre las cuales se construye también pueden presentarse como *patrones arquitectónicos*.

Los patrones arquitectónicos son especificaciones de diseño que ayudan a los programadores a seguir ciertos lineamientos que han sido probados con anterioridad y que han demostrado ser efectivos. En el caso de la tecnología Java existe una documentación extensa acerca de varios tipos de patrones de diseño [Java patterns, www.javasoft.com]. Entre los patrones arquitectónicos, los que más nos interesan son aquellos que trabajan a favor de la aplicación web y en particular para estructurar su presentación.

Las aplicaciones web de hoy en día utilizan arquitecturas que les permiten manejar la combinación de los tres elementos principales que las componen:

Contenido: La información que se puede presentar en forma de texto, imágenes, u otros multimedios dentro de la aplicación.

Presentación: El diseño visual de las interfaces. Esto involucra un esquema de organización del contenido, uso de colores y otros elementos de diseño para incrementar la usabilidad de la aplicación.

Reglas del Negocio: Son las especificaciones propias del negocio tras la aplicación web. El qué hacer y cómo hacerlo. Como resultado de la aplicación de estas reglas se obtienen los contenidos que son combinados con el formato de presentación.

Entre los modelos arquitectónicos que permiten combinar estos tres elementos dinámicamente y de forma estructurada, uno de los más populares es el paradigma Model-View-Controller (MVC). La esencia de este modelo es simplemente la separación entre lógica del sistema y diseño visual, algo enfatizado por la programación orientada a objetos. Por ser una propuesta tan simple, este modelo se ubica en un nivel más alto de abstracción que los demás patrones, y no existen actualmente otros modelos que se puedan considerar alternativas al MVC. Lo que sí existe son diferentes implementaciones del mismo concepto, en donde solo varía la tecnología y la forma de usarlo. El MVC no es nada nuevo, nació con Smalltalk-80, y debido a las necesidades de los sistemas basados en el web, se ha incrementado su popularidad hasta el punto en que casi todas las tecnologías para desarrollo en este ambiente, como Java, PHP, Perl, .Net, Cold-Fusion, etc., lo implementan. La siguiente sección presenta una descripción más a fondo este paradigma.

4.1.1 El Patrón Model-View-Controller (MVC)

El paradigma MVC es muy conocido dentro de las aplicaciones orientadas a objetos pues, como se menciono anteriormente, fue introducido por SmallTalk para el diseño interfaces en los años

setenta. El MVC especifica tres elementos: el modelo, la vista y el controlador (16)

El modelo engloba las reglas del negocio, es un elemento independiente que resuelve por si solo las funcionalidades principales y maneja aspectos de la aplicación web tales como conectividad de datos u otros afines.

La vista se compone de las pantallas que serán presentadas al usuario. Estas se pueden codificar en forma de plantillas de manera que el layout de la aplicación este totalmente independizado de los demás elementos, en términos prácticos esto es decir que el código del layout no estará mezclado con el resto del código del sistema. Esto facilita el desarrollo y mantenimiento pues se puede trabajar sobre el formato sin tener que acceder a la lógica de la aplicación.

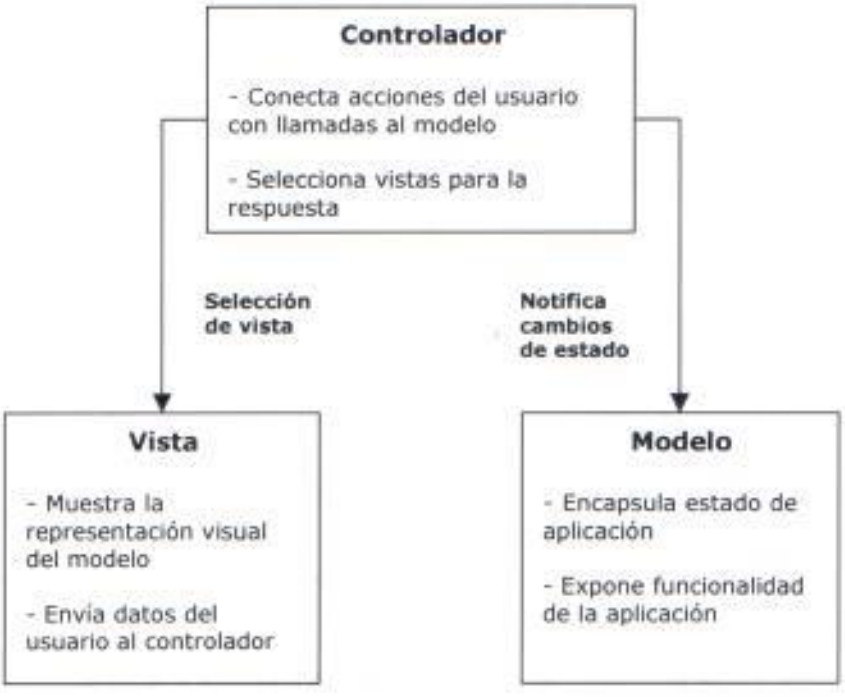
Finalmente ***el controlador*** es el componente encargado de responder a los eventos externos y ensamblar la aplicación, delegando la ejecución al modelo y combinando los resultados con las vistas. La Figura 4.1 muestra un esquema de esta arquitectura.

Existen algunas consecuencias interesantes que se producen como resultado de utilizar este patrón:

- **Re-uso de componentes del modelo**. La separación del modelo y la vista permite el acceso de diferentes módulos al mismo modelo. El diseño, desarrollo, prueba y mantenimiento de los componentes del modelo se facilita, dado que todo el acceso al modelo se efectúa a través de estos.

- **Extensibilidad para nuevos clientes**. Para soportar un nuevo tipo de cliente, simplemente se añade una vista, se informa al controlador y se conecta al modelo ya existente.

- **Mayor Complejidad de diseño**. Este patrón introduce algunos objetos adicionales debido a la separación entre la vista, el modelo y el controlador, como por ejemplo el objeto controlador mismo, objetos que representen acciones que se conecten con lógica del negocio, objetos para manejar la vista, etc.



Fuente: Sun Microsystems, http://www.java.sun.com

Figura 4.1 Las interacciones que ocurren en una aplicación basada en MVC

4.2 Tecnologías Actuales con Soporte para MVC

Hoy en día existen diversos proyectos que dan soporte al patrón MVC en la forma de un marco de trabajo o una librería que permita a los desarrolladores trabajar con esta herramienta. Las Tabla 4.1 muestra algunos de los más populares:

Proyecto	Tecnologías	Descripción
PHP MVC	PHP	Implementación de MV en PHP orientado a objetos
Struts	Java, Servlets, JSP, Tag libraries, Java beans	Implementación de MVC en java, basándose en estándares.
Turbine	Java, Servlets, Velocity, Web Macro	Implementación de MVC utilizando tecnologías alternativas a los estándares.
FuseBox	Cold Fusion	Implementación del concepto MVC en script .cfm

Tabla 4.1 Proyectos para la implementación de MVC

Probablemente la tecnología en donde estos proyectos están más maduros es Java, debido a que estos se valen del API de Java y su auge en el mercado. Ambos son proyectos open-source (código abierto) de la fundación Apache y tienen una comunidad significativa de usuarios y desarrolladores.

No es necesario utilizar una plataforma proveniente de una tercera parte, como las que se menciona en la Tabla 4.1, para aplicar el paradigma MVC en nuestros sistemas. El diseñador se puede ajustar al patrón construyendo los objetos necesarios desde cero. Si bien es cierto esto presenta la ventaja de que no requiere de más software que el que se provee en las distribuciones estándares, la desventaja

en términos de complejidad, confiabilidad y tiempo de desarrollo podría ser grande. Es decir, sin importar cuan compleja sea la implementación que queramos hacer del patrón, su desarrollo implica el consumo de tiempo adicional utilizado en resolver un problema de diseño. Además, proyectos como los antes mencionados tienen años de desarrollo y depuración continua, y normalmente están pensados para solucionar por si solos la mayor parte de las necesidades de desarrollo, por lo que incluyen un conjunto completo de librerías, por ejemplo Turbine, incluye librerías para el manejo de repositorios (*pools*) de objetos, capas intermedias entre bases de datos, envío de correos electrónicos, comunicación con procedimientos remotos, validación de formularios, etc. La ventaja del tiempo de desarrollo ganado así como la disponibilidad de un conjunto completo de funcionalidades, se puede ver contrastada por el tiempo de aprendizaje de la tecnología. Por eso, es importante tomar en cuenta la documentación de la cual estas disponen. En el caso de Turbine y Struts, ambas poseen documentación disponible en línea (http://jakarta.apache.org/turbine, http://jakarta.apache.org/struts), libros publicados (ver en el sitio de Struts) y listas de discusión activas.

En la siguiente subsección se presenta una descripción y comparación de Struts y Turbine, tecnologías que como se mencionó, son basadas en Java y desarrolladas de forma independiente por el proyecto

Jakarta de la Fundación Apache (http://jakarta.apache.org). Adicionalmente, debido a que Java es una tecnología abierta y los proyectos de la Fundación Apache son *open-source*, existe concordancia con nuestros objetivos de mantener bajo costo y alta accesibilidad.

4.2.1 Turbine

Turbine es un marco de trabajo basado en Servlets para el desarrollo de aplicaciones web en Java que permite construir sistemas de forma rápida y segura. Turbine provee una implementación del controlador MVC (ver Figura 4.1) y un mecanismo para el manejo de las vistas y modelos

Turbine puede ser integrado con tecnologías como Velocity, WebMacro, Java Server Pages (JSP), FreeMaker y Cocoon para crear vistas basadas en plantillas dinámicas. Esto permite a los diseñadores trabajar en paralelo con los ingenieros que diseñan el modelo. A continuación se describe en más detalle esta herramienta.

Como se muestra en la Figura 4.2, Turbine esta compuesto de cinco módulos diferentes los cuales sirven un propósito

específico dentro del marco de trabajo de la herramienta. Cada una de estos módulos se describe a continuación.

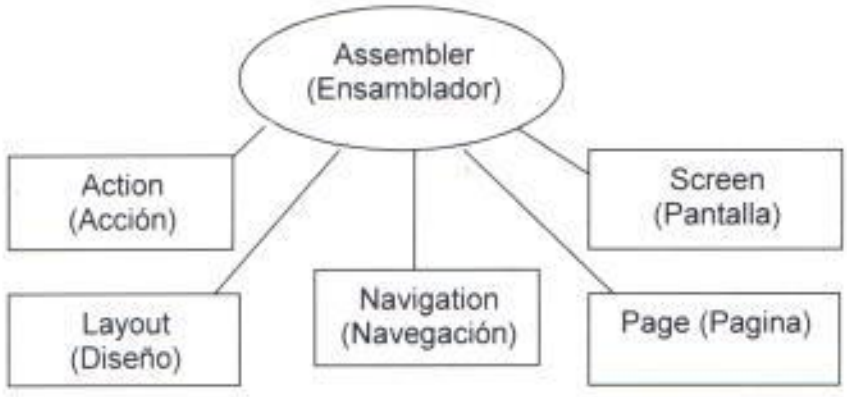


Figura 4.2 Los módulos que componen Turbine

Actions (Acciones)

Un módulo "Action" representa una porción de código que realiza una tarea específica. Por ejemplo, cuando un usuario completa un formulario web, este formulario puede tener campos escondidos, es decir que no son visibles para el usuario, pero que han sido ubicados por el programador para que su información sea enviada junto con el resto de la información ingresada por el usuario. Uno de estos campos podría ser el nombre de la acción que debe ejecutarse para procesar el formulario. El procesamiento generalmente incluye validación de campos y accesos a una base de datos. La página es responsable de ejecutar el "Action" antes que el "Screen" sea

ejecutado. De esta forma, un "Action" puede determinar qué "Screen" se debe ejecutar dependiendo de los resultados.

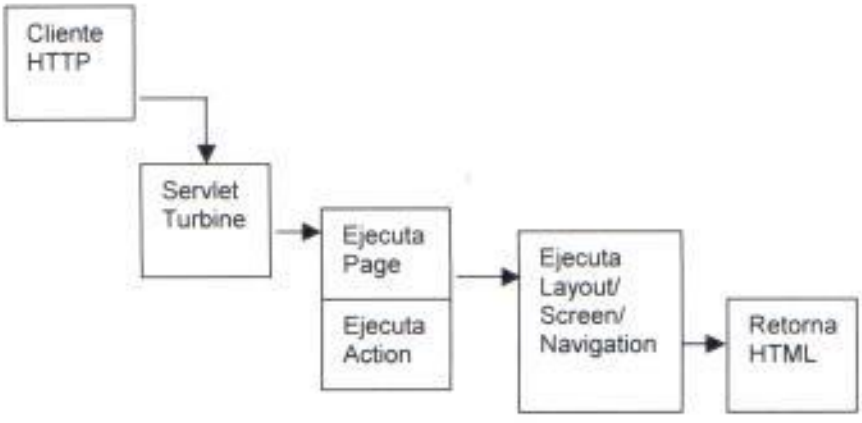


Figura 4.3 Ciclo de ejecución de una aplicación con Turbine

Este modelo permite separar el procesamiento de los datos en componentes que pueden ser re-usados. Por ejemplo, la acción "Logout" puede ser re-usada desde un número de puntos diferentes en el sistema. Esta realiza una sola función y la realiza correctamente. La ventaja de este modelo es que evita ubicar el código para el manejo de los datos de formularios en el Servlet. Esto es útil si se quiere conectar la aplicación web de forma transparente a un modelo empresarial, o a objetos remotos, donde los "Actions" se serían los encargados de establecer la conexión.

Pages (Páginas)

El módulo "Page" es el primer módulo en la ejecución de la página. Este se considera como el módulo que contiene el resto de los módulos que se muestran en la Figura 4.2 (Action, Layout, Screen y Navigation). El módulo "Page" es el encargado de detectar si ha sido definido un "Action" en la petición HTTP. De ser así, este intenta ejecutar la acción. Después de que la acción ha sido ejecutada, este pregunta al objeto Screen por su Layout. "Page" luego intenta ejecutar el objeto Layout que el Screen retornó. Obsérvese que la acción puede modificar qué "Screen" debe ser ejecutado. Los "Screen" además pueden redefinir el Layout que utilizarán, aunque por defecto utilizan el que haya sido especificado como "DefaultLayout" en el archivo de propiedades (properties) de la aplicación.

Screen (Pantalla)

El módulo Screen es esencialmente considerado el cuerpo (body) de la página web. El módulo "Layout" ejecuta el "Screen". Es aquí donde el HTML de la página es generado. Además, es posible llamar código externo aquí, por ejemplo para configurar el contenido de la página.

Navigation (Navegación)

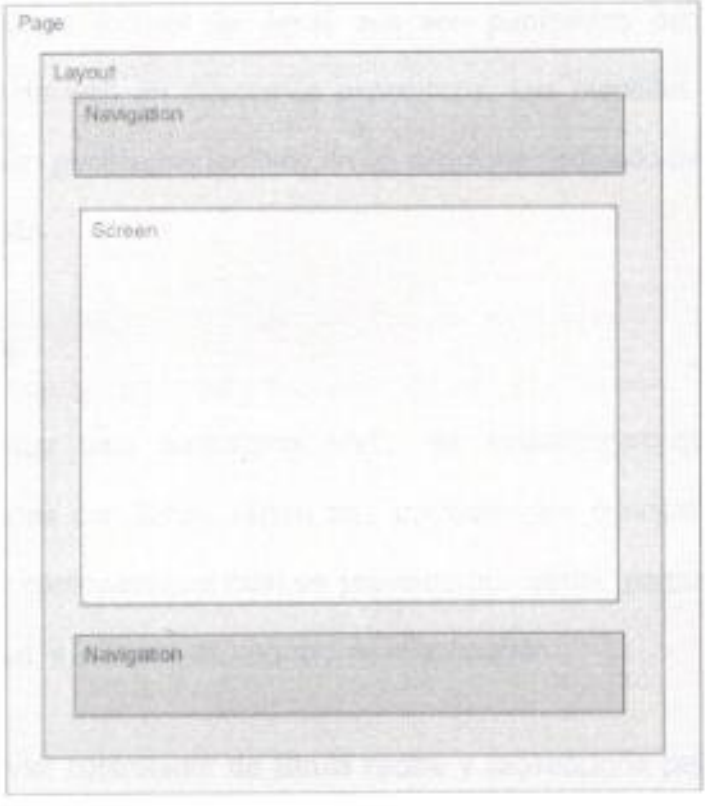
Un sitio web generalmente tiene un esquema de navegación con barras superior e inferior. Esto normalmente se conoce como "header" y "footer" del sitio web. La navegación es ejecutada por el Layout. Puede haber múltiples módulos de navegación que el Layout ejecuta (la barra lateral, superior e inferior de las páginas). Dado que es común para diferentes páginas tener la misma navegación, usualmente se definen "Layouts" para los "Screens" con diferentes navegaciones. Con esto se puede tener múltiples navegaciones que pueden ser incluidas condicionalmente en el Layout.

Layout (Diseño)

El módulo "Layout" es llamado desde el módulo "Page". Este define el diseño físico de la página. Generalmente define la ubicación de los "Navigation" así como la ubicación del cuerpo de la página (o "Screen"). El módulo "Layout" ejecuta el "Screen" para construir la página y los "Navigation" para definir la navegación.

Desde el punto de vista de la encapsulación de un módulo, el esquema de la Figura 4.4 describe como encajan los módulos en conjunto. Como se puede ver, esto tiende a lucir como una

página web diseñada a base de plantillas. Esto no es accidental, ya que el marco de trabajo de Turbine es esencialmente una representación orientada a objetos de los componentes de una página web.



Fuente: Proyecto Jakarta, Fundación Apache

Figura 4.4 Esquema usado por Turbine para descomponer la página web

Desde el punto de vista de desarrollo Turbine representa una enorme ventaja pues permite despreocuparse por la implementación del controlador MVC (ver Figura 4.1) y la generación de código HTML. El desarrollo de la aplicación simplemente consiste en diseñar e implementar los módulos individuales (clases de Java) que son publicados dentro del componte web en directorios específicos. Las plantillas por su parte son publicadas también en un directorio dedicado para este propósito.

4.2.2 Struts

Por estar bajo paradigma MVC, las aplicaciones que son diseñadas con Struts tienen tres componentes principales: Un servlet controlador, el cual es proveído por Struts, paginas JSP (la vista), y la logica de negocio de la aplicación.

El Servlet controlador de Struts recibe y redirecciona peticiones HTTP hacia otros objetos en la aplicación, incluyendo paginas JSP y objetos que sean subclases de "org.apache.struts.action.Action". Cuando se inicializa el controlador (la primera vez que este es solicitado por una petición HTTP) se procesa un archivo de configuración. Este

archivo define (entre otras cosas) los mapeos entre peticiones HTTP y acciones para una aplicación.

El objeto "Action" puede manejar las peticiones para responder a un cliente (usualmente un web browser), o indicar que el control debe ser redireccionado hacia algún otro componente. Al redireccionar, un "Action" puede tener acceso a los métodos del Servlet controlador y enviar JavaBeans hacia otros objetos en la aplicación.

En una aplicación Struts, la mayor parte de la lógica del negocio puede ser representada con JavaBeans. Un "Action" puede utilizar las propiedades de un JavaBean sin necesidad de saber como trabaja este, pues la lógica esta encapsulada para que el "Action" se pueda enfocar en el manejo de errores (validaciones de formularios, manejo de excepciones, etc.) y redirección de peticiones.

Struts utiliza JSP para manejar la producción de HTML. Otros componentes como los "Actions" y los JavaBeans utilizan los archivos JSP que encapsulan la presentación de la aplicación. Las paginas JSP hacen mas sencillo para el desarrollador la

generación de código HTML; sin embargo, internamente en el servidor estos documentos son convertidos en Servlets.

Dado que Struts se ajusta a la especificación estándar del lenguaje y la tecnología web en Java, no puede dejar de incorporar los "Tag Libraries" para incluir funcionalidades prediseñadas en el JSP, que manejen aspectos como internacionalización o manejo de formularios.

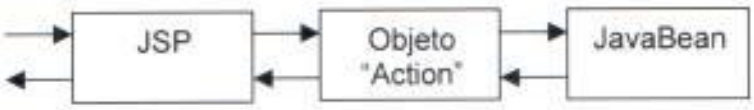


Figura 4.5 Modelo de comunicación entre los tres componentes principales en Struts

La Figura 4.5 muestra la arquitectura básica de la aplicación basada en Struts. En esta figura no aparece el controlador que maneja la aplicación de forma transparente a través de los mapeos especificados en el recurso de configuración.

Struts es una alternativa que se mantiene fiel a las especificaciones del API de Servlets y de JSP producidas por Sun Microsystems (http://java.sun.com/docs/books/jls/). Desde esta perspectiva, posee características que pueden ser beneficiosas en ciertos escenarios, y otras que podrían

entorpecer el desarrollo. Por ejemplo, es más sencillo para aquellos que ya conocen JSP, Java Beans, librerías Tag, y otros elementos que involucra la tecnología estándar de componentes web en Java. Sin embargo, para un desarrollador que no conoce estas tecnologías, la curva de aprendizaje podría ser más larga.

En la siguiente sección se muestra una breve comparación entre los dos marcos de trabajo aquí vistos, y se sustenta la elección de uno de estos para el desarrollo de esta tesis.

4.2.3 Turbine vs. Struts.

La principal diferencia entre estos marcos de trabajo es la orientación tecnológica de fondo con la que fueron desarrollados. Mientras Struts se limita a las especificaciones de las tecnologías producidas por Sun Microsystems, intentando sacarles el mejor provecho, Turbine es un proyecto en donde se pensó desde cero como hacer más fácil y eficiente el desarrollo con MVC.

Turbine utiliza Velocity [http://jakarta.apache.org/velocity], un subproyecto de Jakarta-Apache que provee un lenguaje interpretado muy parecido a PHP para la generación de las plantillas. Esto, aunque no esta contenido dentro de la especificación formal del lenguaje Java, permite a los

diseñadores trabajar de forma más independiente de los analistas y programadores encargados del modelo, ya el vínculo con los objetos de Java se hace a través de un lenguaje sencillo de *script*. A diferencia de esto, Struts utiliza JSP, que debe pasar por un proceso de compilación interno que dificulta el desarrollo. Turbine por su parte permite separar la lógica específica de las páginas en los componentes "Screen".

Existe otra diferencia importante: Struts utiliza un archivo de configuración para indicar el comportamiento del controlador, mientras que Turbine se basa en el concepto de publicar en directorios los "Actions" y plantillas. Es decir, mientras Struts necesita cambios en su configuración para encontrar los componentes requeridos, Turbine lo hace directa y automáticamente utilizando el nombre de los archivos. Esto, como veremos en el siguiente capítulo, es vital para la arquitectura de una aplicación con comportamiento dinámico como la que se plantea en esta tesis, donde el resultado del proceso de razonamiento debe permitir cargar los componentes para construir la aplicación en tiempo de ejecución y sin supervisión de un administrador. Por estos motivos, se ha utilizado Turbine para el desarrollo de la plataforma planteada en esta tesis.

4.3 Sumario.

Hasta aquí hemos revisado las teorías y tecnologías necesarias para plantear un modelo que permita el desarrollo de una aplicación sensitivas al contexto. Hemos revisado el problema de la sensitividad al contexto como un proceso, donde la política de obtención de decisiones en función de los datos de entrada puede ser descrita con los mapas cognitivos difusos. Discutimos además sobre su impacto como herramienta para el manejo del conocimiento organizacional y cognitivo. En este capitulo hemos revisado las tecnologías web sobre las cuales construiremos una propuesta. Se concluye también que Turbine es el marco de trabajo mas propicio para implementar una arquitectura que permita desarrollar aplicaciones adaptativas.

En la siguiente sección se describe cómo estas teorías y tecnologías se acoplan en el diseño de una plataforma que permita construir aplicaciones inteligentes en base a la sensitividad al contexto.

CAPITULO 5

5. MVC y FCM. UNA PLATAFORMA EXTENDIDA PARA APLICACIONES INTELIGENTES

5.1 Diseño de la Solución Híbrida

En este capitulo se describe una propuesta para extender la arquitectura Model-View-Controller (MVC) que soporta la aplicación web con los mapas cognitivos difusos (FCM) vistos en el capitulo 3.

Para extender la arquitectura MVC con FCMs, el componente controlador debe ser capaz de comunicarse con el motor de inferencia del FCM, de forma que la aplicación pueda acceder al comportamiento inteligente codificado en el mapa cognitivo. Por consiguiente, el controlador actuará de acuerdo a la siguiente secuencia:

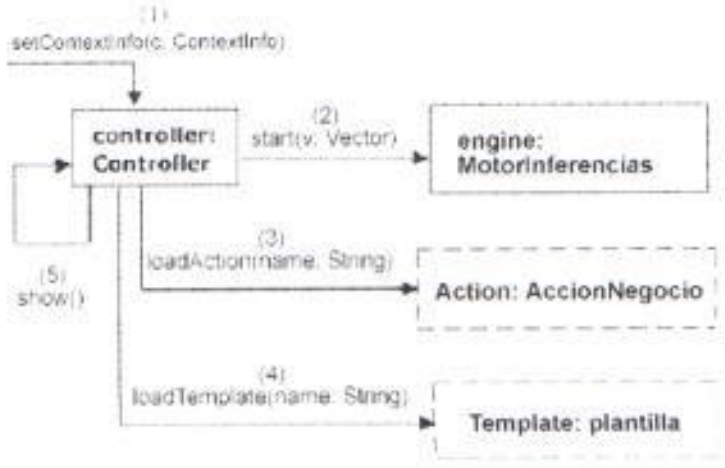
- La información del contexto (datos de la sesión, datos históricos, información del negocio, etc.) es pasada al controlador junto con las peticiones http del cliente.

- El controlador utiliza la información del contexto para crear el estado de entrada del FCM e iniciar el proceso de inferencia.

- El controlador selecciona las acciones y plantillas de diseño apropiadas de acuerdo a las salidas del proceso de inferencia.

- El controlador fusiona las salidas de las acciones con las plantillas de diseño y muestra los resultados al cliente.

A partir de ese esquema, la Figura 6 muestra un grafo de interacción de objetos para una aplicación genérica utilizando la arquitectura MVC extendida con FCMs.



Fuente: (4)

Figura 5.1. Arquitectura MVC extendida con FCMs.

El uso de FCMs para controlar el comportamiento de la aplicación demanda compatibilidad entre las entradas y salidas definidas en el FCM y los elementos disponibles en la aplicación web. Esto es, los nodos iniciales del FCM deben referirse a la información del contexto de la aplicación mientras que los nodos finales deben referirse a acciones disponibles y plantillas de diseño.

En una perspectiva más amplia, este tipo de aplicaciones serán desplegadas probablemente en un ambiente de tres capas, donde el controlador, la lógica del negocio, las plantillas de diseño y el motor de inferencias serán parte de la capa intermedia. La información del contexto será generada a partir de las capas front-end y back-end

como datos de sesión y datos organizacionales respectivamente. La capa back-end también alojará los conceptos del FCM y su estructura, o en otras palabras, el conocimiento organizacional.

Figura 5.2. La aplicación inteligente en un ambiente 3-capas.

Para diseñar los mapas se utilizará el software desarrollado en el Centro de Tecnologías de Información de la ESPOL. Esta herramienta gráfica permite la creación de mapas cognitivos de forma colaborativa. Es decir que muchos expertos pueden describir su conocimiento en línea y enriquecer los resultados finales a través de la discusión. Además permite la combinación de mapas al trabajar bajo el concepto de workspaces (Áreas de trabajo) donde se tienen muchos mapas representando diferentes porciones de un mismo sistema. Otra característica importante de esta herramienta es que nos permite evaluar los resultados de la inferencia y medir el grado de activación de los conceptos a través del uso de colores y otras ayudas

cognitivas. A continuación se muestran algunas vistas que muestran las funcionalidades principales de la herramienta:

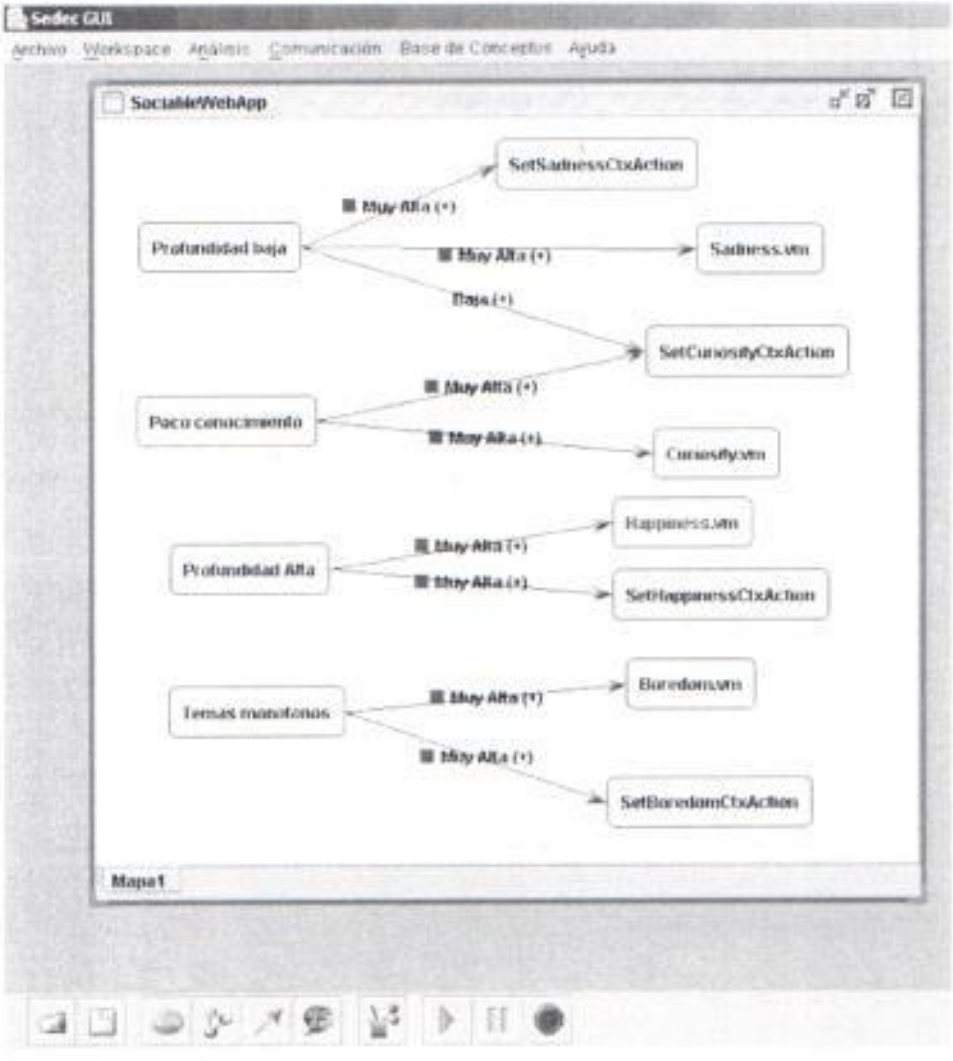


Figura 5.3. Uso de la herramienta para edición de mapas cognitivos difusos

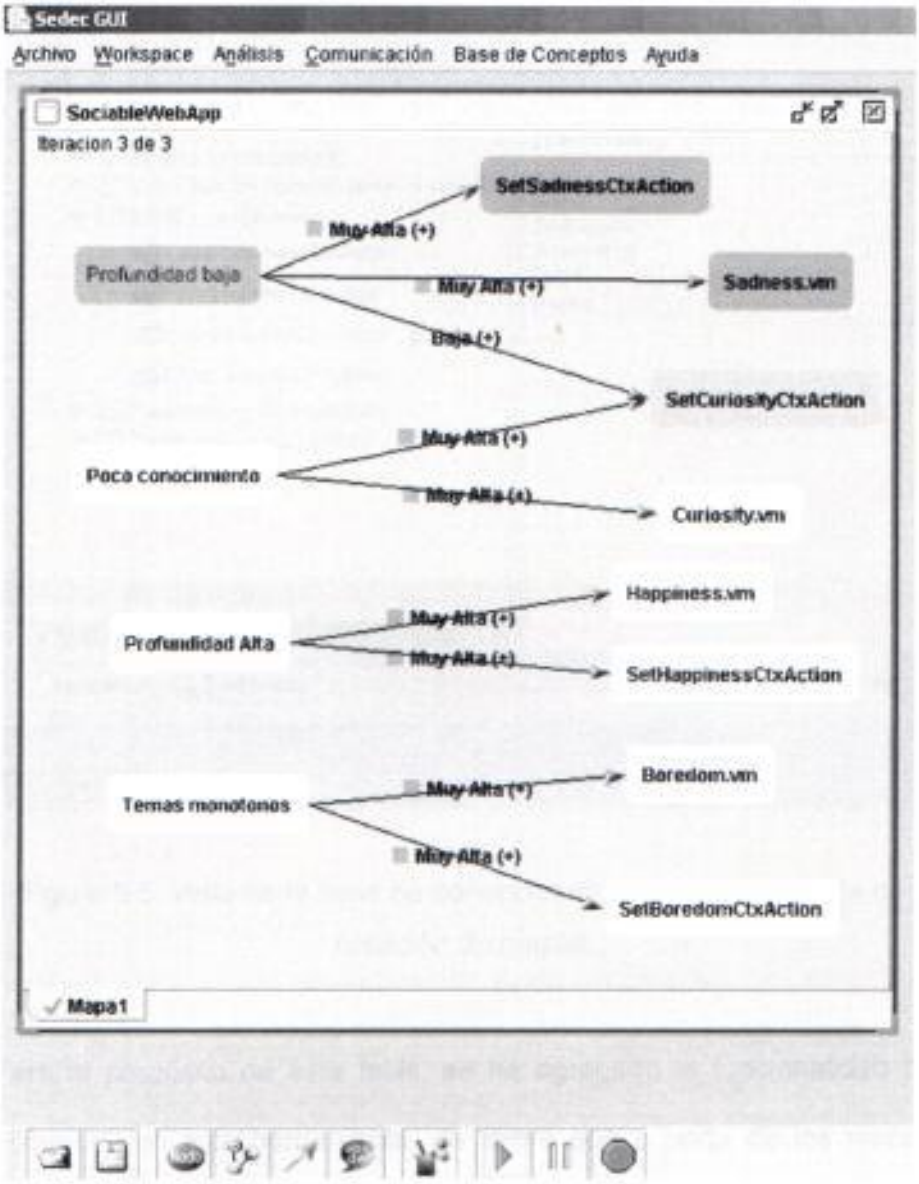


Figura 5.4. Uso de la herramienta para la predicción de los resultados de la inferencia

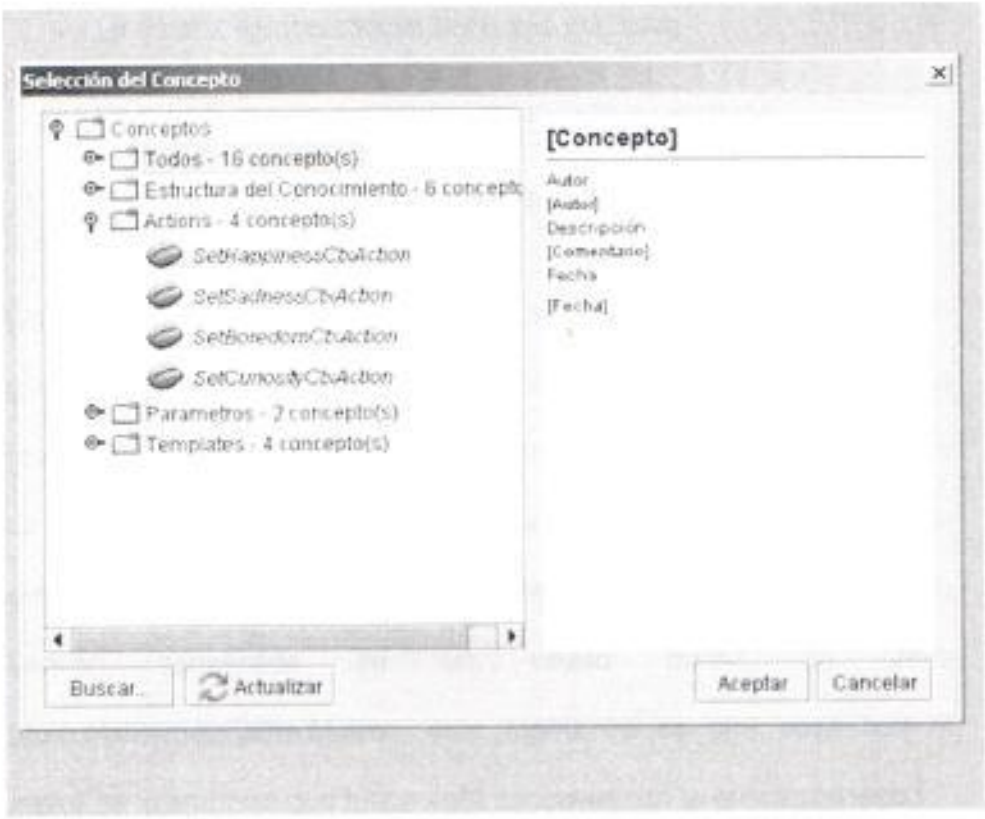


Figura 5.5. Vista de la base de conocimientos en la herramienta de creación de mapas

Para el propósito de esta tesis, se ha agregado la funcionalidad de "exportar" en esta herramienta, de forma que a partir de los mapas cognitivos desarrollados se pueda obtener archivos que luego sean llevados hasta la aplicación web para permitirle acceder a la estructura y datos del mapa cognitivo. En concreto, es necesario exportar tres cosas principales:

- La matriz de conexiones del mapa cognitivo
- Un diccionario de conceptos de entrada en el mapa
- Un vector con los nodos finales del mapa.

Como se menciono en el capitulo 3, el mapa cognitivo, dado que es un grafo, puede ser representado por una *matriz de conexiones* o una matriz de relaciones difusas. Así, a partir de la multiplicación iterativa de un vector de estado inicial con la matriz se puede efectuar el proceso de inferencias (12). Por esto la herramienta exporta una versión serializada de un objeto matriz en Java (com.mathworks.jama.Matrix), esta matriz no es otra cosa que el motor de inferencias que luego será conectado en la aplicación web.

Uno de los obstáculos más importantes para llevar a cabo nuestra visión de diseño, es la traducción de la información contextual en entradas del mapa cognitivo. Para resolver este problema utilizamos un *diccionario de conceptos de entrada en el mapa* (exportado por la herramienta) y un archivo que contiene las reglas que definen los mapeos entre los datos del contexto y los elementos del diccionario. Para instalar estos archivos simplemente se los publica en un directorio llamado "/mappings". El proceso de mapeo se describe en la Figura 5.6:

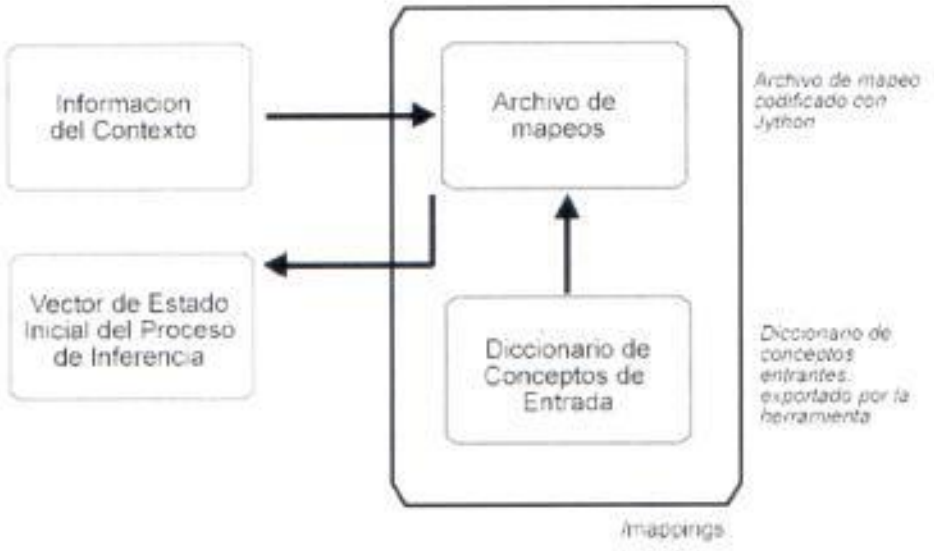


Figura 5.6. Proceso de mapeo de información del contexto en un vector de estado inicial para el motor de inferencias

En la Figura 5.6 se muestra el proceso de mapeo como ha sido resuelto. Contamos con dos elementos claves, el diccionario de conceptos de entrada y el archivo de mapeos. La información del contexto obtenida en la aplicación es transmitida al archivo de mapeos, que utiliza las referencias del diccionario para activar elementos pertinentes en el vector de estado inicial.

Para poder hacer este mapeo se requiere de un lenguaje con la suficiente capacidad de expresar reglas complejas pero manteniendo la flexibilidad de ser interpretado en tiempo de ejecución. En base a estos requerimientos se ha elegido usar Jython, una herramienta en

Java que nos permite programar en Python y combinar código en este lenguaje con programas en Java. Python es un lenguaje interpretado y orientado a objetos. El hecho de ser interpretado le añade a Java un conjunto de facilidades anteriormente inexistentes, como por ejemplo el poder acceder desde código compilado en Java a variables y funciones que se encuentran en archivos interpretados y que contienen código Python. Además, desde estos archivos interpretados se puede acceder a toda clase de objetos de Java. De esta forma se crea una relación sinérgica entre ambos lenguajes, permitiéndole a Python acceder al enorme conjunto de APIs de Java, y permitiéndole a Java contar con la flexibilidad de un lenguaje interpretado.

Usando Jython se ha podido encontrar una solución para definir los mapeos de forma flexible. En la siguiente figura se muestra un ejemplo de un archivo de diccionario:

```
dict['usuario menor de 25 años'] = 3
dict['alto numero de productos en bodega'] = 1
dict['poca participación de usuarios en website'] = 0
dict['comprador frecuente'] = 3
```

Figura 5.7. Ejemplo del diccionario de conceptos de entrada

El diccionario de conceptos de entrada contiene un objeto PyDictionary de Jython, un tipo de arreglo común en los lenguajes interpretados, donde las claves son cadenas de caracteres. En este diccionario las claves son los nombres de los conceptos en el mapa cognitivo y el valor un número entero que indica la posición del concepto en la matriz de conexiones.

Este diccionario es utilizado por el archivo de mapeos para hacer referencia a las posiciones de los conceptos en la matriz de conexiones, de forma que se pueda crear el estado inicial del proceso de inferencias activando las posiciones de los conceptos seleccionados en el mapeo.

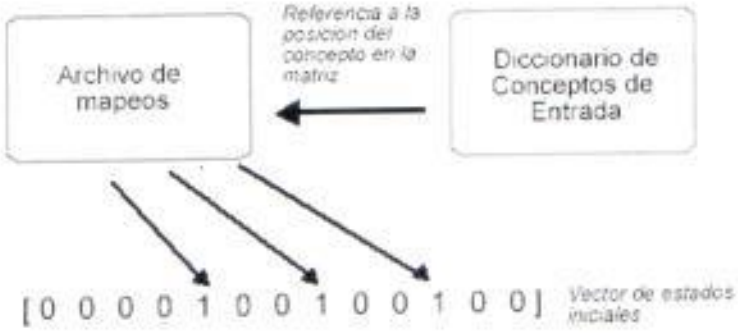


Figura 5.8 Creación del vector de estado inicial

La Figura 5.8 ilustra el proceso de creación del vector de estados iniciales a partir del archivo de mapeos y del diccionario de conceptos de entrada.

La Figura 5.9 muestra como luce el archivo de mapeos. Las reglas que convierten la información del contexto en estados del mapa cognitivo son definidas de forma sencilla a través de expresiones if – else codificadas en Python. El archivo de mapeos recibe un objeto llamado *contextInfo*, una tabla de hash con la información del contexto expresada como un par clave-valor. Además tiene una referencia al objeto *initialState*, el vector sobre el cual se activan los conceptos de entrada y que finalmente será retornado al programa en Java. El vector originalmente tiene un valor de 0.0 en todas sus posiciones, luego el archivo de mapeos asigna un valor de 1.0 en la posición del concepto que debe ser activado.

```
If contextInfo.get("edad") < 25 :
        initialState.set (0, dict['usuario menor de 25 años'], 1.0)

If contextInfo.get("numero de visitas") > 10 :
        initialState.set(0, dict['usuario frecuente'], 1.0)

If contextInfo.get("contribuciones ") /
   contextInfo.get("numero de visitas ") < 0.4 :
    initialState.set(0, dict['poca participación de usuarios en website'], 1.0)
```

Figura 5.10. Ejemplo de un archivo de mapeos.

Hasta aquí se ha descrito detalladamente el proceso a través del cual se obtiene un vector de estados inicial a partir de los datos del contexto. El siguiente paso consiste en efectuar el proceso de inferencia utilizando la matriz de conexiones del mapa cognitivo que describe el comportamiento de la aplicación.

Como se dijo anteriormente, la matriz de conexiones es exportada directamente por la herramienta como una versión serializada de un *objeto matriz de Java. Este objeto es publicado en la aplicación web al* ubicarlo en un directorio designado para este propósito, por defecto este directorio es "/engine". De esta forma una vez obtenido el vector inicial a través del proceso de mapeo, se procede a efectuar el proceso de razonamiento a través del API desarrollado para el proyecto SEDEC en el Centro de Tecnologías de Información de la ESPOL. Este API contiene una clase llamada *Engine* la cual expone un método de clase llamado *start* que permite efectuar el proceso de inferencia. Este método recibe la matriz de conexiones y el estado inicial.

En la Figura 5.11 se puede observar la especificación de la clase "cti.metis.engine.Engine". El método *start* es un método polimórfico que puede retornar una pila con la historia del proceso de

razonamiento o simplemente un vector con el estado final. Es esta última forma del método la que nos interesa.

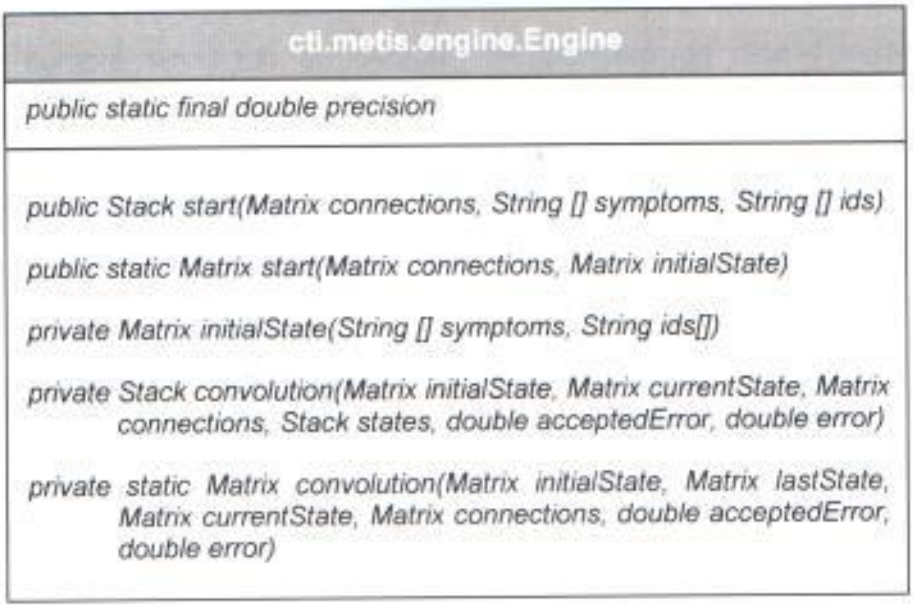


Figura 5.11 Definición de la clase cti.metis.engine.Engine

Este objeto encapsula totalmente el proceso de inferencia del que se habló en el capitulo 3, de forma que la aplicación web solo le entrega la matriz de conexiones y el vector inicial del mapa. El objeto retorna los nodos activados, que representan las acciones ha ejecutarse.

Estas acciones pueden ser de tres tipos:

- Ejecución de "Actions"
- Selección de plantillas de diseño
- Selección de parámetros

Para poder identificar a cual de estos tres elementos pertenece el nodo final se utilizan los nombres de los conceptos, considerando que por convención las clases que representan acciones siempre tendrán un nombre terminado en "Action", las plantillas de diseño serán archivos con una extensión especifica y los parámetros tendrán la forma "nombre = valor". Una vez obtenidos los nombres de estos elementos se procede a ejecutarlos.

A continuación se sumariza el algoritmo general que permite procesar la información del contexto y generar decisiones a partir de esta:

1. Se adquieren los datos del contexto
2. Se efectúa el mapeo utilizando el archivo de mapeos y el diccionario de nodos entrantes y se obtiene el vector de estado inicial.
3. Se realiza el proceso de inferencia y se obtiene el vector de estados finales.
4. Se obtiene los nombres del "Action" y la plantilla de diseño, más una lista de los parámetros seleccionados.
5. Se agregan los parámetros, se ejecutan los objetos "Action" y se muestra la información utilizando la plantilla.

En los siguientes párrafos discutiremos como se ha logrado encajar esta solución en un modelo de clases que se ajuste al paradigma MVC.

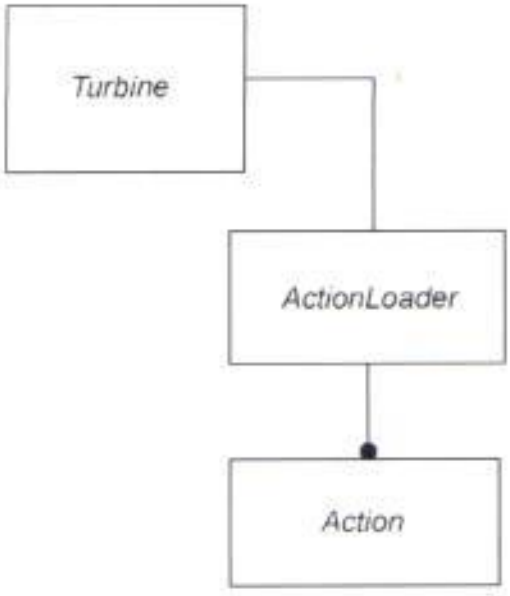


Figura 5.12 Clases de Turbine para el manejo de "Actions"

En la Figura 5.12 se muestra una parte del diagrama de clases del marco de trabajo de Turbine. En este, la clase *Turbine* es el controlador que recibe las peticiones HTTP. Según lo que indique el parámetro "action" en la petición, el controlador invoca al *ActionLoader* que se encarga de cargar en tiempo de ejecución un objeto tipo *Action*. Hay que notar que el vínculo entre *ActionLoader* y *Action* es dinámico. En tiempo de compilación estas clases no poseen vínculos

entre si, pero en tiempo de ejecución el *ActionLoader* añade uno o más objetos tipo *Action* al ambiente de ejecución de la maquina virtual.

Para poder realizar el proceso de mapeos e inferencias como se describe en los pasos 1 – 5 en la página 83, hemos necesitado adaptar la secuencia de llamadas entre los objetos del *framework* para poder situar el procesamiento necesario en un *Action* que sirva como punto de entrada a los demás componentes.

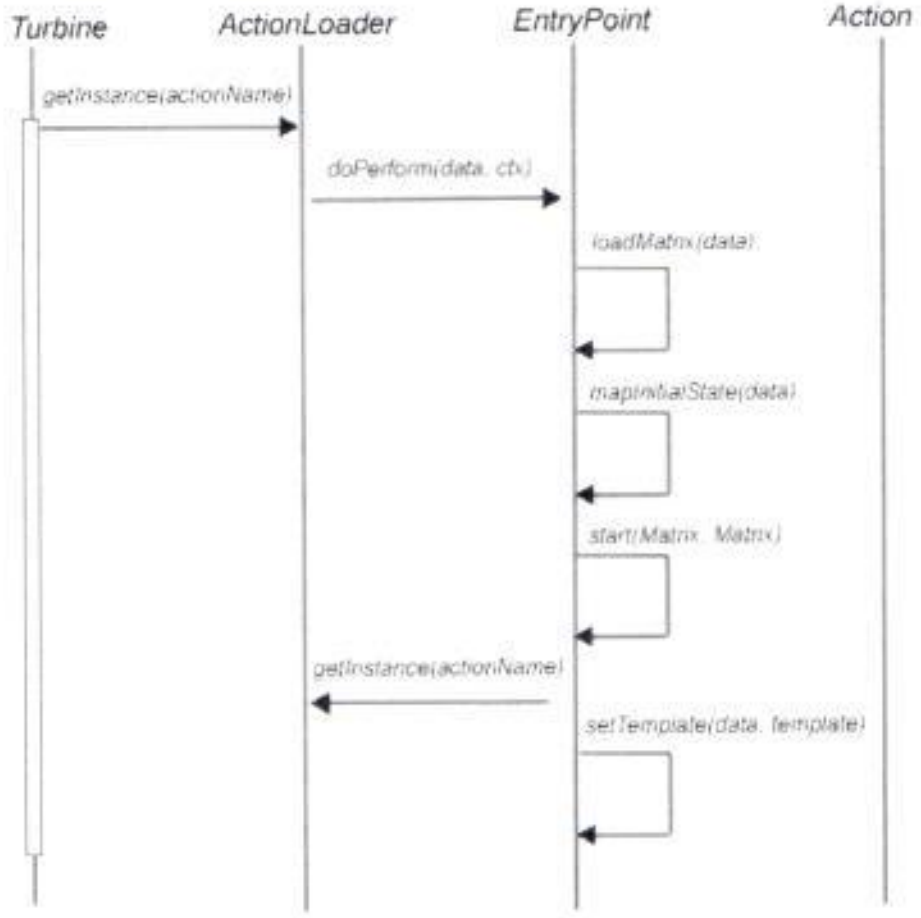


Figura 5.13. Diagrama de interacción de objetos.

En la Figura 5.13 se muestra el diagrama de interacción de objetos que permite satisfacer el algoritmo propuesto en este capitulo. El objeto *EntryPoint* es el punto de entrada en el escenario en donde se realiza el algoritmo de mapeo e inferencia. El objeto *EntryPoint* recibe la información del contexto y carga la matriz de conexiones a través del método loadMatrix(RunData data). Luego realiza el mapeo a través de la función mapInitialState(RunData data), esta función crea el vector de estado inicial para el proceso de inferencia. El proceso de inferencia se lleva a cabo en start(Matrix, Matrix), que produce el vector de estados finales. A partir de estos estados se obtienen los *Actions*, *Templates* y parámetros. Para llamar al action seleccionado se invoca nuevamente a *ActionLoader*, y para mostrar un *Template* se invoca la función setTemplate(RunData data, template).

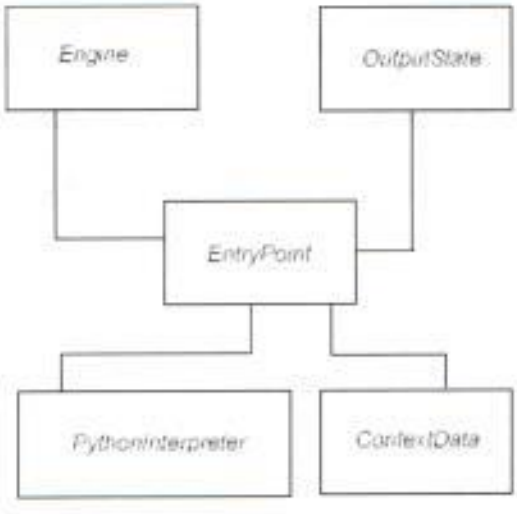


Figura 5.14. EntryPoint y los objetos de apoyo.

En la Figura 5.14 se muestran las relaciones en las que se apoya el objeto EntryPoint para cumplir su funcionalidad. *ContextData* es un objeto que encapsula la información del contexto. Se muestra como una *interfase* que puede ser implementada para envolver y procesar cualquier tipo de información del contexto que sea pertinente (datos del usuario, datos del negocio, estado de la aplicación, etc.). El *PythonInterpreter* es el objeto del API de Jython con el cual se interpreta el archivo de mapeos. Este objeto se inicializa con la información del contexto contenida en *ContextData* y el archivo de mapeos. El objeto *Engine* es el motor de inferencias que se va a inicializar con la matriz exportada desde el mapa cognitivo y que servirá para obtener el estado final a partir del estado inicial que arroja el *PythonInterpreter* luego de interpretar el archivo de mapeos. Finalmente OutputState es un objeto utilitario que permite obtener los nombres del *Action*, *Template* y parámetros que resultan del estado final del proceso de inferencia. Para esto intercepta el estado final con el vector de nodos finales exportado a partir del mapa cognitivo.

5.1 Perspectiva Operacional

Para culminar este capitulo se presenta un análisis de la eficiencia en términos de tiempo de ejecución del diseño propuesto. El rendimiento de la aplicación se encuentra sujeto al rendimiento de cada una de sus

partes constituyentes, sean estas bases de datos, generación de HTML en el servidor, lógica del negocio, etc. Adicionalmente en este caso contamos con el tiempo de ejecución del proceso de inferencias en el FCM. Esta nueva variable es función directa del nivel de complejidad del mapa cognitivo y puede reducir, al menos en teoría, el rendimiento global de la aplicación. Para hacerlo más explicito, el rendimiento en tiempo de ejecución de la aplicación puede expresarse por la formula:

$$\tau(n) = T_S + T_G + T_B + \psi(n) \qquad (5.1)$$

Donde $\tau(n)$ es el tiempo de ejecución total, n es el numero de nodos en el mapa cognitivo, T_S , T_G ,T_B son los tiempos de ejecución del servidor web, generación de html por parte de la aplicación (en este caso los Actions y Templates) y sentencias en la base de datos, respectivamente. Todos independientes de n. $\psi(n)$ es el tiempo de ejecución del proceso de inferencias en el mapa cognitivo. Depende de las valencias que se den a los pesos de las relaciones entre nodos en el FCM. Por ejemplo para valores {-1, 1}, donde dos estados son considerados iguales si todos sus elementos tienen igual signo, el tiempo de ejecución es proporcional a 2^n (14).

De acuerdo a las pruebas reportadas en (3), el tiempo que le toma al proceso de convolución llegar al estado estable, si bien esta dado por una función exponencial es de lento crecimiento, como se muestra en la Figura 5.15

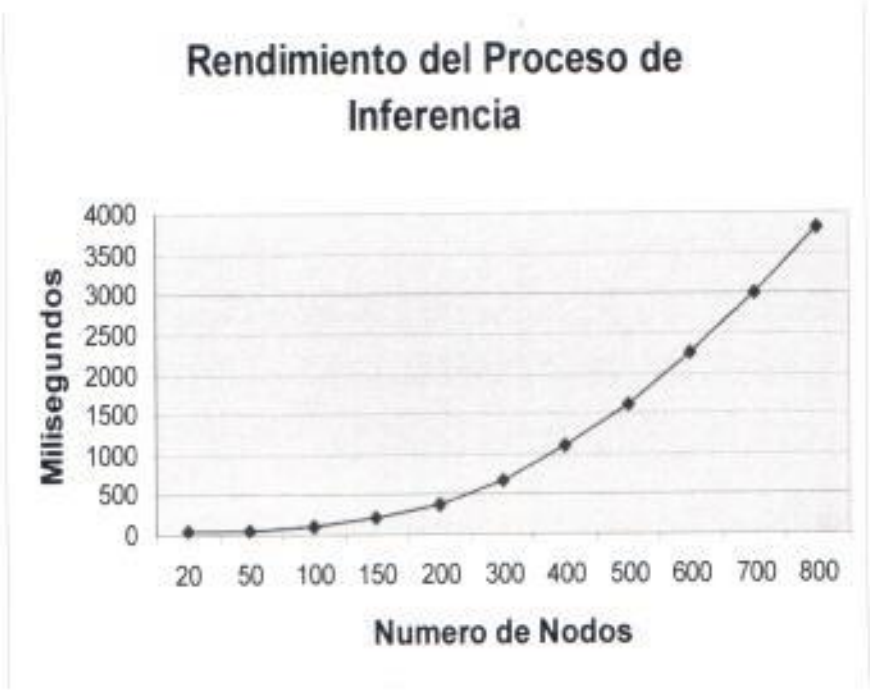


Figura 5.15. Rendimiento del proceso de inferencia

Fuente: (4)

Según estas pruebas realizadas en un computador Pentium II de 450 Mhz un mapa cognitivo de 100 nodos apenas representará menos de ¼ de segundo en tiempo de ejecución. Además como también se menciona en (4) un mapa cognitivo de alrededor de 100 nodos o mas

quizás represente un modelo muy complejo como para ser puesto en producción o un mal diseño que deba ser revisado. Por esto el énfasis en la necesidad de ingenieros del conocimiento para dar mantenimiento y controlar la corrección de los modelos, ya no para servir de intermediarios entre los expertos y el sistema.

CAPITULO 6

5. PROTOTIPO DE UNA APLICACIÓN WEB SOCIABLE

En este capitulo se describe una implementación prototipo de una aplicación Web sociable con el fin de poner a prueba la flexibilidad de la plataforma. Además este capítulo debe servir como guía para nuevas implementaciones que hagan uso del esquema propuesto en el capítulo anterior.

En la siguiente subsección se hará una introducción a las aplicaciones Web sociables y posteriormente se describirá el prototipo en base a su especificación funcional y ejemplos.

6.1 Introducción a las Aplicaciones Sociables

Lo social y lo emocional están altamente interconectados. Para algunos investigadores las emociones entran en juega tan pronto como consideramos a los individuos en interacción con su entorno

social. Para otros, las emociones están en el corazón mismo de lo que significa ser sociable. Esta alta interdependencia ha tenido eco en la inteligencia artificial, particularmente en el área de los agentes socialmente inteligentes (3), donde se han hecho grandes esfuerzos en el modelamiento de las emociones para interacciones sociales.

¿En que pueden las emociones artificiales contribuir a la interacción entre los artefactos y los humanos? ¿Cuales son los principales retos que implica su diseño? Intuitivamente, podemos pensar en los diferentes roles que la emociones pueden jugar en las interacciones sociales entre humanos y maquinas. Se mencionará algunos de ellos (3):

- *Transmitir intencionalidad.* La gente necesita entender el comportamiento observado en sus contrapartes sociales – humanas o artificiales – como resultado de causas o intenciones que permitan formar una explicación coherente de las observaciones. Esta coherencia es necesaria para interpretar relaciones pasadas o presentes, hacer predicciones y establecer expectativas acerca de comportamientos futuros. Emociones y personalidades son usualmente postulados como causas de comportamiento y como fuentes de intenciones cuando se intenta explicar el comportamiento de humanos o

animales. Uno puede esperar entonces que estas sean usadas de la misma forma cuando se interactué con artefactos. Los sistemas autónomos pueden además usar expresiones para transmitir necesidades o intenciones a los humanos.

- *Despertar emociones.* En la misma forma que la emoción de otra persona despierta respuestas emotivas en humanos, las emociones transmitidas por un artefacto pueden ser usadas con el mismo propósito, buscando respuestas que encajen con el estado emocional del artefacto (un asistente de vuelo que intenta despertar un estado de alerta en el piloto) o que le sirvan de instrumento (un robot expresando tristeza debido a su inhabilidad de completar una tarea puede recibir ayuda de un humano 'conmovido')

- *Confort humano.* Artefactos capaces de expresar emociones y adaptar su interacción al estado emocional de sus 'socios' pueden hacer sentir más confortables a los humanos durante la interacción. Una razón obvia es que esta interacción esta apuntada a satisfacer las necesidades emocionales de los humanos. Otra razón importante es que las emociones artificiales pueden hacer al artefacto más creíble, en un sentido percibido como más 'cercano' o similar a nosotros.

- *Comunicación mejorada.* La expresión emocional, siendo un elemento clave en la comunicación no verbal, puede ayudar a los artefactos a hacer la comunicación con humanos menos costosa en términos cognitivos. Si los artefactos emocionales pueden alcanzar un nivel de sofisticación lo suficientemente alto en un futuro, podemos soñar con un nivel de comunicación mas profundo, que por ejemplo permita interpretar nuestras pistas contextuales y emociones sutiles, *entendiendo* lo que queremos decir, no solo lo que decimos.

Consideremos ahora los principales retos en el diseño de artefactos emocionales con interacciones sociales.

- *Qué emociones implementar.* La mayoría de los proyectos que implementan emociones en artefactos han optado por un subconjunto de emociones básicas, como tristeza, felicidad, dolor, placer, etc. Estas son fácilmente reconocibles por los humanos y son sencillas de sintetizar en las maquinas. Sin embargo, cuando se desea lograr una interacción mas avanzada con un robot por ejemplo, es necesario representar expresiones mas variadas y por tanto la demanda es más grande para el diseñador.

- *Inspiración de la naturaleza.* ¿Hasta que grado podemos tomar como inspiración los modelos y teorías de las emociones humanas y animales para diseñar artefactos emocionales? ¿Es esto deseable y posible? Conclusiones y directrices muy útiles pueden obtenerse a partir del análisis de los estudios de las emociones humanas. Se puede obtener además herramientas para resolver problemas específicos y realizar pruebas. Sin embargo, se debe considerar que el realismo extremo puede ser nocivo al colocar expectativas muy altas en los humanos, un robot es un robot, y un humano es un humano, por esto los artefactos deben de posicionarse como herramientas útiles y sus capacidades sociales deben estar destinadas a mejorar la interacción con los humanos.

No se puede dejar de lado las complicaciones que tiene el modelar emociones de forma que estas sean creíbles o al menos que sean comunicativas tal como se lo proponen los diseñadores. Esto dependerá del tipo de artefacto del que estemos hablando, usualmente en esta área de la inteligencia artificial se trata de robots capaces de simular emociones a través de expresiones faciales o comportamientos específicos. La coherencia del comportamiento es otro aspecto muy importante, en tanto que los humanos esperamos

ver las emociones como consecuencia de un conjunto de causas o intenciones.

Sobre este tema, se cree que las emociones expresadas por el artefacto son más creíbles si corresponden a un sistema de emociones 'profundo' embebido en este. Es decir que no solo basta por ejemplo con lograr objetos antropomórficos sino además con proveer una base que permita desarrollar comportamientos emocionales coherentes.

Si queremos que los artefactos emocionales vayan más allá de lo superficial y 'contengan' las emociones en cierto sentido, se deben encontrar formas de anclar las emociones de manera apropiada al ambiente dinámico y a los miembros del entorno humano.

Una forma de anclar las emociones es adherir al artefacto módulos o componentes que representen de forma explicita las emociones. Estos componentes actúan como una caja negra que responde con emociones simuladas dependiendo de ciertas entradas.

En otro acercamiento, las emociones están enraizadas en el artefacto, entrelazadas con su mecanismo perceptual y de acción. Esto implica que el agente realizara su propia categorización de emociones. Normalmente este tipo de implementaciones utilizan simulaciones de

mecanismos naturales (por ejemplo redes neuronales adaptativas) y en teoría permitirían al artefacto desarrollar su propio sistema de valores y categorías.

Los artefactos sociables, usualmente robots, deben incorporar ciertos mecanismos para poder comportarse de la forma esperada. Uno de estos elementos la motivación social. Las emociones pueden ser consideradas como un control del comportamiento de segundo orden que monitorea el cumplimiento de las metas del sistema motivacional Otro elemento más avanzado, y aun lejos de nuestro alcance, es la teoría de la mente. Esto es, la habilidad de entender el comportamiento de otros dentro de un marco intencional. Los artefactos que implementen una teoría de la mente permitirían interacciones que hasta ahora no son posibles entre maquinas y humanos, los robots podrían aprender de la observación y comunicarse sin necesidad de un lenguaje artificial. Finalmente, otro fenómeno que debe ser considerado es la empatía y simpatía. La habilidad de los seres humanos de conectarse con los estados emocionales de los demás y adoptarlos en diferentes grados.

Estos elementos representan grandes retos para la ciencia informática actual y su desarrollo es definitivamente de primera importancia para la sociedad. En la siguiente subsección se tratará como llevar los

conceptos de las aplicaciones sociables al Web utilizando el marco de trabajo descrito en los capítulos anteriores.

6.2 Especificación Funcional

La aplicación sociable que se describe en este capítulo sirve para demostrar la utilidad de las propuestas de esta tesis y vuelve visibles puntos muy interesantes en lo que respecta a las aplicaciones sociables. Además, como se menciono anteriormente es un punto de partida para posibles implementaciones más avanzadas.

La posibilidad de 'anclar' el sistema emotivo usando los mapas cognitivos es algo novedoso y útil, en tanto que este representa a la vez el modelo de emociones y el motor que produce las respuestas a partir de las entradas del ambiente.

El mapa cognitivo que se encuentra atado al controlador de la aplicación Web funciona como una caja negra en la cual el sistema de emociones no es visible para el espectador. Este mapa es definido por los expertos en la lógica que rige el comportamiento del sistema, que no son necesariamente los programadores.

Otro aspecto muy importante es que en las aplicaciones basadas en ventanas y por tanto en las aplicaciones Web, que podrían ser

considerados como una variación menos dúctil de las aplicaciones con ventanas, la representación de las emociones es mucho más complicada que con robots en los cuales se puede simular gestos faciales. En las interfaces de software tradicional se debe usar pistas contextuales para sugerir estados de ánimo: colores, tipos de letra, cambios de comportamiento, gráficos, etc., son las herramientas que *se deben administrar para construir las emociones artificiales.*

El ejemplo de la aplicación sociable es apropiado para demostrar el potencial de la plataforma debido a que requiere el manejo de un *modelo complejo de comportamiento que se manifiesta en la selección* de diferentes expresiones. En teoría además este conjunto de expresiones es continuo, es decir que el modelo debe encargarse de "discretizar" las posibles expresiones o salidas.

Para el prototipo que se presenta en este capitulo se ha construido una aplicación que simula un sistema de recomendaciones para los clientes de una farmacia. El sistema utiliza la información acerca de los perfiles de los usuarios para "decidir" que productos recomendar. Adicionalmente, el sistema adapta tanto su presentación como contenidos de acuerdo a las características del usuario, siguiendo así los principios de las aplicaciones sociales.

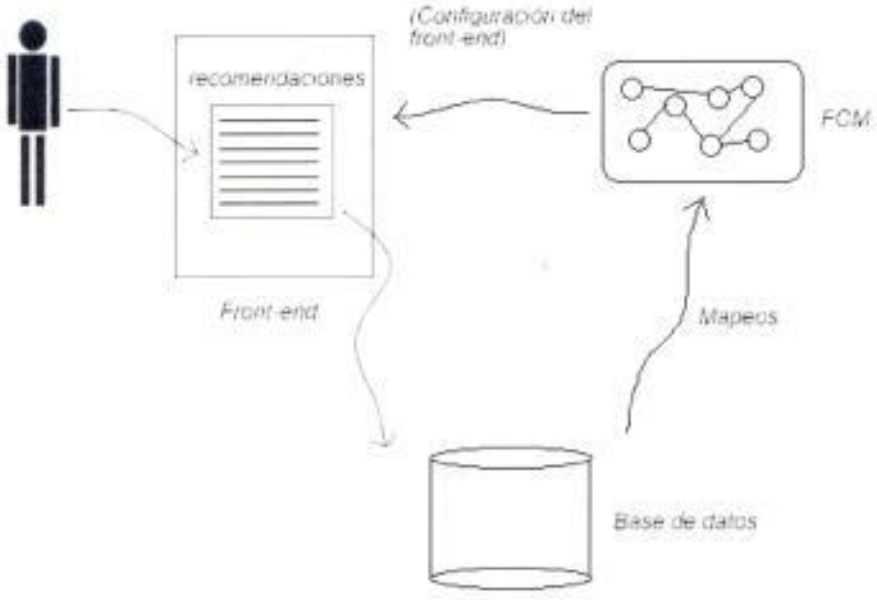


Figura 6.1 Esquema de la aplicación Web sociable basada en la plataforma propuesta.

En la Figura 6.1 se muestra el esquema del funcionamiento de esta aplicación en base a la plataforma propuesta en el capitulo 5. En este esquema la interfase permite al usuario registrarse e ingresar al sistema con un nombre de usuario y una clave. Durante el registro, el usuario debe especificar su edad y sexo. Estos datos luego serán transformados por el archivo de mapeos en la información del contexto usada por el mapa cognitivo para iniciar el proceso de inferencias. Las

salidas de este proceso indicarán que categorías de productos recomendar y como adaptar la infernase del usuario.

A continuación se describe cada uno de los componentes del prototipo de forma más detallada:

- ***Front-end* (Interfase con el usuario).** La interfase con el usuario esta compuesto por una plantilla de Velocity y dos objetos Screen que servirán para configurar la página de acuerdo al sexo del usuario. Este par de objetos determinarán aspectos como el color y el contenido de la página que se enviará al cliente. El front-end permite ingresar un nombre de usuario y un password, en base a éstos el sistema levantará los datos del contexto para iniciar el razonamiento con el mapa cognitivo. Finalmente se mostrara la pantalla con las características seleccionadas por el estado final del mapa.

Figura 6.2 Pantalla principal del sistema.

La figura 6.2 muestra la pantalla principal del sistema. Esta muestra un formulario de registro y otro de ingreso.

- **Mapa Cognitivo**. El mapa cognitivo modela el comportamiento central de la aplicación. Básicamente interconecta los posibles estados de las entradas con conclusiones lógicas, y estas con las correspondientes acciones a realizarse en el sistema.

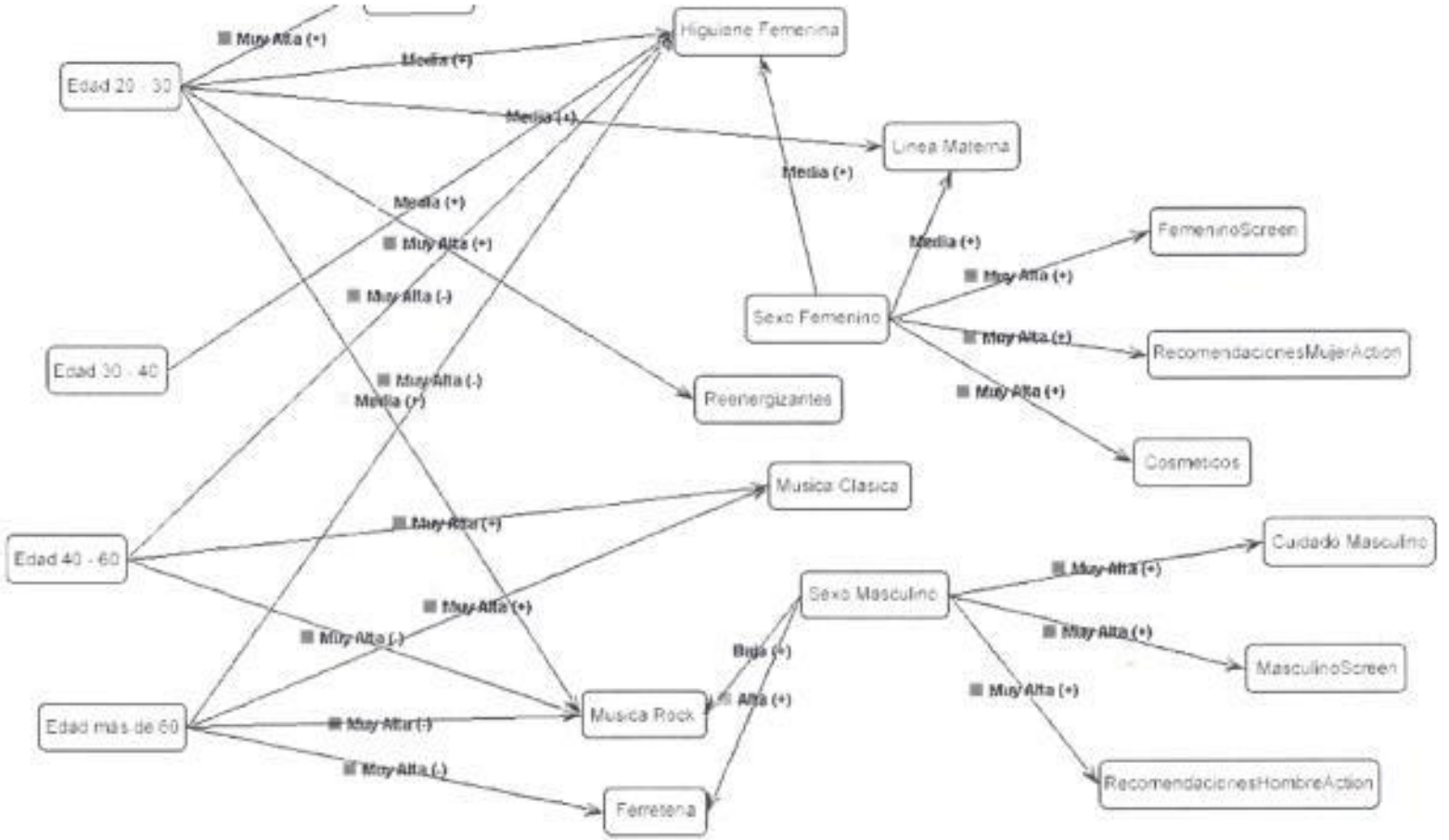


Figura 6.3. Mapa cognitivo con la lógica del negocio

En la Figura 6.2 se muestra el mapa cognitivo que define el comportamiento del sistema. A partir de los conceptos de entrada *Edad 20 – 30, Edad 30 – 40, Edad 40 – 60, Edad más de 60, Sexo Femenino, Sexo Masculino*, se concluye acerca de las categorías de productos que son más adecuadas para el usuario. Por ejemplo si es el usuario es hombre entre 20 y 30 años, muy probablemente preferirá artículos de ferretería que si es mujer entre 40 y 60.

- **Archivo de Mapeos.** En el caso de este prototipo la información del contexto es bidimensional, es decir existen dos parámetros (sexo y edad) que son utilizados en el archivo de mapeos para activar los nodos del mapa cognitivo. En la medida en que se aumenten dimensiones se vuelve más extenso el archivo de mapeos y el comportamiento de la aplicación se vuelve menos predecible.

  El archivo con el diccionario de conceptos de entrada se muestra en la Figura 6.4:

```
dict['Edad 20 - 30'] = 0
dict['Edad 30 - 40'] = 1
dict['Edad 40 - 60'] = 2
dict['Edad mas de 60'] = 3
dict['Sexo Masculino'] = 4
dict['Sexo Femenino'] = 5
```

Figura 6.4. Diccionario de conceptos de entrada

Con los conceptos mostrados en la figura anterior se realiza define el proceso de mapeos según como se muestra en la Figura 6.5

```
if contextInfo.get("edad") < 30 :
        initialState.set(0, dict['Edad 20 - 30'], 1.0)

elif contextInfo.get("similarity") < 40 :
        initialState.set(0, dict['Edad 30 - 40'], 1.0)

elif contextInfo.get("similarity") < 60 :
        initialState.set(0, dict['Edad 40 - 60'], 1.0)

else:
        initialState.set(0, dict['Edad mas de 60'], 1.0)

if contextInfo.get("sexo") == 'M' :
        initialState.set(0, dict['Sexo Masculino'], 1.0)

else:
        initialState.set(0, dict['Sexo Femenino'], 1.0)
```

Figura 6.5 El archivo de mapeos del prototipo.

En la información del contexto contenida en el objeto *contextInfo* existen dos variables: sexo y edad. En función de estas se escoge el estado inicial del mapa cognitivo. El valor de edad es un número entero mientras que el sexo puede tener el valor M o F, que indican Masculino y Femenino respectivamente.

Hasta aquí se ha descrito en detalle el funcionamiento del prototipo a partir de la descripción de cada uno de sus componentes – front-end, mapa cognitivo y archivo de mapeos. Cualquier aplicación que se construya sobre este marco de trabajo tendrá una estructura similar, compuesta de los mismos componentes pero con diferentes valores para cada uno.

Para el prototipo se ha simulado una farmacia y su lógica de negocio. En esta existen categorías de productos, productos y usuarios, que se encuentran almacenados en una base de datos. La Figura 6.6 muestra un diseño de tablas para el ejemplo.

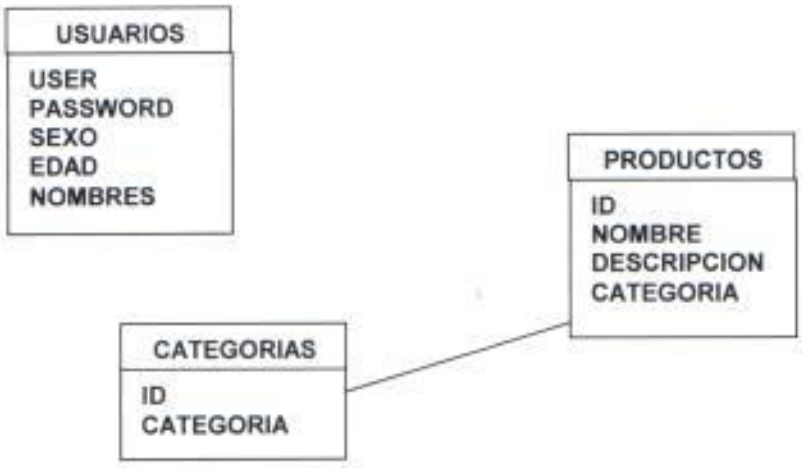


Figura 6.6. Diseño de tablas para el ejemplo

Las categorias de productos que se encuentran en el sistema son las que se muestran en la Tabla 6.1

Categorías de Productos
Cuidado Acné
Línea Materna
Cuidado de Bebe
Higiene Femenina
Cosméticos
Cuidado Oral
Reenergizantes
Ropa Hombres
Ferretería
Tecnología
Ropa Mujeres
Electrodomésticos
Música Rock
Música Clásica
Cuidado Masculino

Tabla 6.1 Categorías de Productos

La Tabla 6.2 muestra los productos almacenados en el sistema con sus respectivas categorías:

Productos	Categorías
Crema Limpiadora	Cosméticos
Crema Antiarrugas	Cosméticos
Juego de Blush y Sombras	Cosméticos
Tratamiento para cabello tinturado	Cosméticos
Disco de Limp Biskit	Música Rock
Disco de Juanes	Música Rock
Palm Pilot Phillips	Tecnología
Laptop Toshiba	Tecnología
Licuadora de 5 velocidades	Electrodomésticos
Batidora de Mano	Electrodomésticos
Sanduchera	Electrodomésticos
Kit de herramientas	Ferretería
Kit de Accesorios para su Auto	Ferretería
Bebida StronGate para Deportistas	Reenergizantes
Afeitadora GrossHair	Cuidado Masculino

Tabla 6.2 Productos con sus respectivas categorías

En base a esta información el sistema construirá la página web que será presentada al usuario, con las recomendaciones de productos que mejor se ajusten a las categorías activadas por el mapa cognitivo. En la siguiente sección veremos ejemplos de cómo se comporta el sistema frente a diferentes perfiles de usuarios.

6.3 Ejemplos

Se han ingresado cuatro usuarios con diferentes características para mostrar como el sistema adapta su comportamiento según el perfil de cada uno. En la siguiente tabla se muestra la información de cada usuario.

Usuario	Edad	Sexo
Otto Cordero	24	M
Sandra Bullock	35	F
Steven Spielberg	55	M
Jennifer López	28	F

Tabla 6.1 Usuarios y sus respectivos perfiles

Los productos recomendados para cada uno de estos perfiles se muestran a continuación:

Otto Cordero (24, M):

Kit de Herramientas, Kit de Accesorios para el Auto, Disco de Limp Biskit, Disco de Juanes, Afeitadora GrossHair.

Sandra Bullock (35, F):

Crema Limpiadora, Crema Antiarrugas, Juego de Blush y Sombras, Tratamiento para cabello tinturado.

Steven Spielberg (55, M):

Kit de Herramientas, Kit de Accesorios para el Auto, Afeitadora GrossHair.

Jennifer López (28, F):

Crema Limpiadora, Crema Antiarrugas, Juego de Blush y Sombras, Tratamiento para cabello tinturado, Bebida StrongGate para deportistas

En estos ejemplos se puede observar que de acuerdo a la lógica contenida en el mapa cognitivo de la Figura 6.3, las recomendaciones varían drásticamente dependiendo del sexo del usuario. En el caso de diferencias de edades hay productos que se mantienen como por ejemplo los relacionados a la ferretería en el caso de los hombres y los cosméticos para las mujeres. Para el usuario Otto Cordero, se sugieren como recomendaciones dos discos de música moderna que no aparecieron para el usuario Steven Spielberg. A su vez, para el usuario Jennifer López se sugirió una bebida para deportistas que no fue mostrada para Sandra Bullock.

En las siguientes Figuras se muestran las pantallas para dos de los usuarios. En estas se puede observar que no solo varían las recomendaciones sino también el aspecto de estas páginas.

Figura 6.7 Página mostrada al usuario Otto Cordero

Figura 6.8 Página mostrada al usuario Sandra Bullock

Estos ejemplos demuestran que es posible modelar la lógica del negocio usando los mapas cognitivos. Esta idea es importante pues el uso de grafos representa un gran salto sobre la barrera sintáctica de los lenguajes formales usados para representar la lógica del negocio. Además, es posible codificar la inteligencia del sistema en los mapas cognitivos, permitiéndole a éste adaptar su comportamiento de acuerdo al contexto en que se encuentre.

CONCLUSIONES Y RECOMENDACIONES

En el transcurso de esta tesis se han propuesto los fundamentos para el desarrollo de aplicaciones Web con comportamiento inteligente sobre la base de la sensitividad al contexto. La estrategia se ha centrado en utilizar un patrón arquitectónico de diseño orientado a objetos para subdividir la aplicación en pequeños componentes y luego combinarlos y ejecutarlos en tiempo ejecución según la respuesta del mapa cognitivo. Este mapa cognitivo es además una herramienta para administrar el conocimiento y puede ser usado para representar las reglas del negocio de una empresa de forma jerárquica y descentralizada.

La aplicación prototipo desarrollada esta basada enteramente en herramientas de código abierto, más especificamente en Turbine, un proyecto de la fundación Apache para dar soporte al desarrollo de aplicaciones Web bajo el esquema MVC. Una de las principales características de este software y por la cual ha sido escogido para el prototipo, es la capacidad de detectar objetos ubicados en directorios durante el tiempo de ejecución. De esta forma los vínculos entre los componentes del sistema no están definidos durante la compilación sino durante tiempo de ejecución.

Al ser Turbine una herramienta cuyo licenciamiento no plantea restricciones sobre su uso o modificación, la solución aquí propuesta mantiene un bajo costo y una alta flexibilidad para ser mejorada según las necesidades. Una de las innovaciones que presenta esta tesis es la combinación de ciertos principios de inteligencia artificial con las herramientas de código abierto que se han mencionado.

En conjunto el prototipo es una plataforma que permite desarrollar aplicaciones con cierto grado de autonomía en el Web. Para esto los desarrolladores agregarían las Acciones y Pantallas requeridas, tal como si estuvieran trabajando con Turbine, pero estas serían ejecutadas desde un punto de entrada que "decidiría" que componente ejecutar de acuerdo al mapa cognitivo que se haya diseñado. Otro de los resultados de esta tesis es la interfaz gráfica que permite diseñar los mapas cognitivos y exportar la matriz de conexiones para que esta sea usada en la aplicación Web.

Los ejemplos mostrados en el Capitulo VI ilustran la flexibilidad de la plataforma desde un punto de vista importante: la capacidad de reaccionar frente a indicadores continuos. Esto significa en la práctica que la aplicación podría adaptar su comportamiento según el resultado de funciones que midan valores reales del ambiente o del dominio, y no simplemente de criterios que puedan tener una o dos alternativas como se da hoy en día.

Uno de los aspectos importantes en la evaluación del prototipo es la eficiencia en términos de tiempo de ejecución. Las pruebas realizadas demuestran que el proceso de inferencias, que en principio aparece como el potencial "cuello de botella", si bien es cierto esta sujeto a una relación exponencial entre el número de conceptos en el mapa y el tiempo de ejecución, no representa una carga mayor pues esta relación crece muy lentamente lo que hace que en la practica el procesamiento se vuelva imperceptible para el usuario.

Sin embargo, la inicialización de la matriz de conexiones a partir del archivo exportado desde la interfaz grafica podría causar demoras al ejecutar por primera vez el punto de entrada de la aplicación. Por esto, nuevas versiones mejoradas del prototipo deberían considerar ubicar la matriz de conexiones bajo la envoltura de un servicio de Turbine. Los servicios de Turbine exponen la funcionalidad de componentes reutilizables administrando su propio ciclo de vida. Esto permitiría manejar más eficientemente la deserialización e inicialización de los objetos requeridos.

Otra de las potenciales mejoras al sistema, aunque mucho más ambiciosa, es darle a los mapas cognitivos la capacidad de aprender y auto-modificarse. Para este propósito se podría utilizar la información contenida en los archivos de *log* del sistema y aplicar algoritmos de detección aprendizaje para modificar la estructura del mapa o los valores de los vínculos entre

conceptos. La principal dificultad a enfrentar, y lo cual vuelve este problema muy complejo, es que los criterios bajo los cuales se evalúa la corrección semántica del mapa son básicamente cualitativos, por lo que la intervención humana es primordial.

Otra potencial mejora a este trabajo es la inclusión de condiciones lógicas (AND, OR, etc.) a la estructura del mapa. De esta forma se podrían construir modelos más complejos que permitan reflejar de forma más precisa la lógica del negocio tras la aplicación Web. Además se podría especializar los nodos del mapa cognitivo para que estos representen no solamente Acciones, Pantallas y Parámetros, sino también llamadas a Métodos, Excepciones, grupos de parámetros, Layouts (Refiriéndose al elemento Layout de Turbine), Navegaciones (Navigations en Turbine), servicios, etc.

Finalmente, cabe resaltar que el uso de grafos para modelar el comportamiento de un sistema, tal como se ha propuesto en esta tesis, podría servir para alivianar la carga que representa expresar la lógica del negocio en una aplicación de tres capas. Típicamente, esta lógica reside en la capa intermedia como un modelo dinámico de objetos, o en la capa de datos como procedimientos nativos del manejador de base de datos. Esta última es una alternativa muy común en el medio por lo que no es extraño ver bases de datos con siete mil o diez mil *store procedures*, sacrificando capacidad de mantenimiento por velocidad de ejecución. La utilización de

grafos tales como los mapas cognitivos, podría ser de mucha utilidad para encontrar un punto medio siempre y cuando se pueda subdividir el sistema en pequeñas funcionalidades que luego puedan ser integradas o llamadas durante la ejecución. En esta línea la computación orientada a componentes podría permitir encapsular las funcionalidades que ahora se encuentran dispersas en la base de datos a través de grupos de objetos que trabajan bajo un contrato y con un ciclo de vida. El mapa cognitivo entonces representaría una federación de componentes bajo una lógica del negocio.

Como se puede apreciar, esta tesis promueve implicaciones que van más allá de los objetivos planteados y de cierta forma se alinea con las iniciativas actuales para conseguir sistemas más fáciles de desarrollar y de mantener, así como más capaces de adaptarse por si solo a los diferentes usuarios y contextos.

ANEXO I

DOCUMENTACION DEL CODIGO FUENTE

Paquetes	
odin.context	Clases e Interfaces para el manejo de la información del contexto
odin.modules.actions	Acciones de Turbine que sirven como punto de entrada para la aplicaciones
odin.util	Objetos utilitarios

Paquete odin.context

Sumario de Interfaces	
ContextData	Esta es una interface generica que define el contrato que deben cumplir los objetos que representen los datos del contexto.

Sumario de Clases	
KbContextData	Implementación del objeto ContextData para manejar los articulos ingresados y obtener el grado de similaridad.

odin.context

Interface ContextData

Todas las clases conocidas que implementan:
KbContextData

public interface **ContextData**

Esta es una interfase genérica que define el contrato que deben cumplir los objetos que representen los datos del contexto.

Versión:
1.0
Autor:
Otto Cordero

Sumario de Métodos

void	**addData**(java.lang.String name, java.lang.Object value) añade un dato mas al Hashtable
java.util.Hashtable	**getData**() obtiene los datos del contexto como un Hashtable
void	**process**() efectua el procesamiento que sea necesario sobre los datos

Paquete odin.modules.actions

Sumario de Clases

AbstractEntryPoint	Es la superclase del punto de entrada para el sistema.
EntryPoint	Este es el punto de entrada para las aplicaciones sensitivas al contexto.

odin.modules.actions

Clase AbstractEntryPoint

java.lang.Object
|
+--org.apache.turbine.modules.Assembler
|
+--org.apache.turbine.modules.Action
|
+--org.apache.turbine.modules.ActionEvent
|
+--org.apache.turbine.util.velocity.VelocityActionEvent
|
+--org.apache.turbine.modules.actions.VelocityAction
|
+--**odin.modules.actions.AbstractEntryPoint**

Subclases Directas Conocidas:
EntryPoint

---

public abstract class **AbstractEntryPoint**
extends org.apache.turbine.modules.actions.VelocityAction

Es la superclase del punto de entrada para el sistema. Este Action toma los datos del contexto y se encarga de iniciar el proceso de inferencia para poder seleccionar otras componentes del framework

Autor:
Otto Cordero

Sumario de Atributos

(package private) Jama.Matrix	**connections**
(package private) ContextData	**contextData**
(package private) java.util.Vector	**endnodes**
(package private) Jama.Matrix	**initialState**
(package private) java.util.Properties	**properties**

Campos heredados de org.apache.turbine.modules.ActionEvent
BUTTON, BUTTON_LENGTH, LENGTH, METHOD_NAME_LENGTH, METHOD_NAME_PREFIX

Sumario de Constructores

AbstractEntryPoint()

Sumario de Métodos

protected OutputState	**getOutputState**() obtiene los nodos de salina activados a partir del vector de estado final
private void	**loadMatrix**(org.apache.turbine.util.RunData data) Carga la matriz de conexiones exportada por la herramienta de creacion de mapas.
protected Jama.Matrix	**mapInitialState**(org.apache.turbine.util.RunData data) mapea los datos del contexto al estado inicial del proceso de inferencias
protected void	**performAction**(java.lang.String actionName, org.apache.turbine.util.RunData data, org.apache.velocity.context.Context ctx) este método ejecuta el doPerform del Action seleccionado luego del proceso de inferencia.
protected abstract void	**setContextData**(org.apache.turbine.util.RunData data) Setea la implementación especifica de los datos del contexto
private Jama.Matrix	**startEngine**() obtiene el del proceso de inferencia com un vector final (Matriz)

Métodos heredados de org.apache.turbine.modules.actions.VelocityAction
doPerform, doPerform, getContext, perform, setTemplate

Métodos heredados de org.apache.turbine.util.velocity.VelocityActionEvent
executeEvents

Métodos heredados de org.apache.turbine.modules.ActionEvent
executeEvents, formatString

Métodos heredados de class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

odin.modules.actions

Clase EntryPoint

java.lang.Object
|
+--org.apache.turbine.modules.Assembler
|
+--org.apache.turbine.modules.Action
|
+--org.apache.turbine.modules.ActionEvent
|
+--org.apache.turbine.util.velocity.VelocityActionEvent
|
+--org.apache.turbine.modules.actions.VelocityAction
|
+--odin.modules.actions.AbstractEntryPoint
|
+--**odin.modules.actions.EntryPoint**

---

public class **EntryPoint**
extends AbstractEntryPoint

Este es el punto de entrada para las aplicaciones sensitivas al contexto. si se desea utilizar el motor de inferencias para seleccionar automaticamente el comportamiento de la aplicación, este es el action que debe ser llamado primero.

Autor:
Otto Cordero
Ver además:
AbstractEntryPoint

Sumario de Atributos

private java.lang.String	**action** Action que sera seleccionado
private java.util.List	**parameters** Parametros que seran seleccionados
private java.lang.String	**realPath** Ruta real en el servidor donde se alojan los archivos
private java.lang.String	**template** Template que sera seleccionado

Campos heredados de odin.modules.actions.AbstractEntryPoint
connections, contextData, endnodes, initialState, properties

Campos heredados de org.apache.turbine.modules.ActionEvent
BUTTON, BUTTON_LENGTH, LENGTH, METHOD_NAME_LENGTH, METHOD_NAME_PREFIX

Sumario de Constructores

EntryPoint()

Sumario de Métodos

private void	**createFile**(org.apache.turbine.util.RunData data) crea un archivo temporal con el contenido del articulo ingresado
void	**doPerform**(org.apache.turbine.util.RunData data, org.apache.velocity.context.Context context) el método doPerform es el que realiza la funcioalidad basica de todo Action internamente, este inicia el proceso de inferencia, obtiene los resultados y procede a ejecutarlos
protected void	**setContextData**(org.apache.turbine.util.RunData data) Inicializa la implementación usada del objeto ContextData y la ubica en el contexto del contenedor de Servlets

Métodos heredados de odin.modules.actions.AbstractEntryPoint
getOutputState, mapInitialState, performAction

Métodos heredados de org.apache.turbine.modules.actions.VelocityAction
doPerform, getContext, perform, setTemplate

Métodos heredados de org.apache.turbine.util.velocity.VelocityActionEvent
executeEvents

Métodos heredados de org.apache.turbine.modules.ActionEvent
executeEvents, formatString

Métodos heredados de java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Paquete odin.util

Sumario de Clases	
OutputState	este objeto engloba el conjunto de nodos finales seleccionados despues del proceso de inferencia.

odin.util

Clase OutputState

java.lang.Object
|
+--**odin.util.OutputState**

public class **OutputState**
extends java.lang.Object

este objeto engloba el conjunto de nodos finales seleccionados despues del proceso de inferencia. Es capaz de recibir un vector de estados final, y obtener las plantillas, acciones y parametros a ejecutar.

Autor:
Otto Cordero

Sumario de Atributos

(package private) java.util.Vector	**endnodes** nodos finales del mapa cognitivo
(package private) Jama.Matrix	**finalState** vector de estado final

Sumario de Constructores

OutputState(Jama.Matrix finalState, java.util.Vector endnodes)
crea un nuevo objeto OutputState

Sumario de Métodos

java.lang.String	**getAction**() Revisa los nodos finales activados del mapa cognitivo y obtiene el Action que se ha seleccionado.
java.util.List	**getParameters**() Obtiene una lista de paramtros a partir de los nodos activados del mapa cognitivo.
java.lang.String	**getTemplate**()

	Obtiene el template a ejecutar a partir de los nodos finales activados del mapa cognitivo.

Métodos heredados de.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

BIBLIOGRAFIA

1.- ROBERT AXELROD, Structures of Decision, The Cognitive Maps of Political Elites. Princeton University Press, New Jersey, 1996.

2.- BROWN, B., BOVEY, J. Y CHEN, X. Context Aware Applications: From the Laboratory to the Marketplace, IEEE Personal Communications, Octubre 1997.

3.- LOLA CAMANERO, Building emotional artifacts in social worlds: Challenges and Perspectives. Emotional and Intelligent II: The tangled Knot of Social Cognition. AAAI Press, 2001.

4.- CORDERO Y PELÁEZ. Dynamic Business Intelligence, Automated Decision Making with Fuzzy Cognitive Maps. BIS 2002, Poznan, Polonia, 2002.

5.- CORDERO Y PELÁEZ. Context-Aware Web Applications handled by Expert Systems. BITWorld 2002, Guayaquil, Ecuador, 2002.

6.- CORDERO Y PELÁEZ. Fuzzy Numbers for the improvement of Causal Knowledge Representation in Fuzzy Cognitive Maps. National Conference of Artificial Intelligence 2002, Edmonton, Canada, 2002.

7.- DAVIS J., BONNELL R., Modeling Semantic Constraints with Logic in the EARL Data Model. Fifth International Conference on Data Engineering, pp. 226-233, 1989.

8.- D. DUBOIS AND H. PRADE, Fuzzy relation equations and causal reasoning. Fuzzy Sets and Systems, pp 119--134, 1995.

9.- E. HISDAL, The IF THEN ELSE statement and interval-valued fuzzy sets of higher type: Int. J. Man-Machine Studies, pp 385-455, 1981, Vol 15

10.- NICHOLAS KASSEM (Editor), Designing Enterprise Applications with the Java(TM) 2 Platform (Enterprise Edition), Addison-Wesley, 2000.

11.- GEORGE J KLIR. AND BO YUAN FuzzySets and Fuzzy Logic: Theory and Applications. Prentice Hall, 1995.

12.- BART KOSKO, Fuzzy Associative Memory Systems. Fuzzy Expert Systems. CRC Press, 1992.

13.- BERNARD LIAUTAUD WITH MARK HAMMOND. e-Business Intelligence, turning information into knowledge into profit. McGraw Hill, 2000.

14.- C. E. PELÁEZ, A Fuzzy Cognitive Map Knowledge Representation for Failure Modes and Effects Analysis. University of South Carolina, 1994.

15.- C. E. PELÁEZ AND JOHN B. BOWLES. Using Fuzzy Cognitive Maps as a System Model for Failure Modes and Effect Analysis. Joint Conference on Information Sciences, 1994.

16.- S. SAUER, G. ENGELS, MVC-Modeling Support for Embedded Real-Time Systems, Object-Oriented Modeling of Embedded Real-Time Systems (OMER), pp. 11-14, 1999.

17.- MARK A. SMITH, Customer Intelligence – Still a Long Way to Go. Intelligent Enterprise, Marzo 2001.

18.- C. D. STYLIOS, P. GROUMPOS, Fuzzy Cognitive Maps: A model for intelligent supervisory control systems. Computers in Industry, pp 229-238, Elsevier Science B.V., 1999.

19.- E. TURBAN, J. LEE, D. KING Y HM CHUNG, Electronic Commerce: A Managerial Perspective, Prentice Hall, 2000.

20.- L. A. ZADEH, The concept of a linguistic variable and its application to approximate reasoning - I. Information Sciences, pp 199-249, 1975.

21.- ZASLAVSKY, ARKADY, Enabling Awareness in Dynamic Mobile Agent Environments, ACM Symposium on Applied Computing, SAC'2000, Italia, Marzo, 2000.

22.- HANS J. ZIMMERMANN, Fuzzy Sets Decision Making and Expert Systems. Kluwer Academic Publishers, 1987.