Casa abierta al tiempo

UNIVERSIDAD AUTÓNOMA METROPOLITANA

Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Maestría en Ciencias de la Computación

Sistema de Aprendizaje Inteligente con Objetos de Aprendizaje "ProgEst"

Tesis para obtener el grado de Maestra en Ciencias de la Computación.

Presenta: María de Lourdes Sánchez Guerrero

Asesor Interno: Dr. en C. Ana Lilia Laureano Cruces

Asesor Externo: M. en I. Martha Mora Torres

México D.F., Julio de 2009

Dedicatorias

A mis padres, a mis hermanos y hermanas, por su apoyo incondicional en todos los proyectos que he emprendido en mi vida.

A David, mi esposo por su amor, apoyo y paciencia en todo momento.

Reconocimientos

A la Dr. Ana Lilia Laureano por su paciencia, *su confianza, su tiempo y espacio, por compartir conmigo su conocimiento. Agradezco su gran apoyo en la dirección de esta tesis.*

A la M. en I. Martha Mora por compartir sus conocimientos, recomendaciones y apoyo en el diseño de esta tesis.

A mis amigas y colaboradoras Ana Lilia González, Dulce María Rosales, Luz María Cabrera por su apoyo a largo de esta etapa de mi vida.

Al Dr. Tomás Miklos, la Dra. Mariem Henaine y a el M. en C. José Raymundo Lira por su apoyo en la lectura, comentarios y recomendaciones al trabajo.

Resumen "Sistema de Aprendizaje Inteligente con Objetos de Aprendizaje (ProgEst)"

Los entornos de aprendizaje están cambiando. Los nuevos escenarios plantean desafíos técnicos y pedagógicos a los que las instituciones de educación superior deben responder. Los roles de los profesores, alumnos y personal de apoyo deben adaptarse a los nuevos entornos. En este caso el objetivo principal radica en conjuntar las nuevas tecnologías con la inteligencia artificial.

El siguiente trabajo describe el análisis y diseño de un SAI ubicado en programación estructurada. La organización de este SAI se basa en OA.

Se desarrolla un SAI, que basa el proceso de enseñanza – aprendizaje en un motor de inferencia inspirado en el proceso tutorial humano y consta de nueve elementos; 1) interés, 2) deseo, 3) ayuda, 4) estrategias cognoscitivas y operativas, 5) interrupción, 6) renuncia, 7) aprendizaje, 8) tiempos inactivos, 9) error. Dichos elementos se relacionan en una matriz causal que permite ver la causalidad de cada uno de ellos con los demás. Además de estos nueve elementos se agregan dos más que permiten enriquecer el modelo del estudiante al considerar el estilo de aprendizaje que tendrá un impacto directo en la interfaz, y la motivación interna de estudio que permitirá elegir las estrategias operativas de acuerdo a ésta última. El motor de inferencia está representado con mapas cognitivos difusos. Este último no es parte del desarrollo de este trabajo.

Por otro lado se diseña el currículo utilizando un grafo genético, el cual apoyado en una arquitectura multiagente, implica a tres agentes expertos cada uno en un sub-dominio. Los sub-dominios están representados por los objetivos instruccionales. Lo anterior permite elaborar un proceso de enseñanza–aprendizaje basado en escenarios ligados a cada uno de estos sub-dominios, que a su vez permiten un manejo más detallado de errores.

Los objetivos instruccionales implicados en este trabajo, de acuerdo a la taxonomía de Bloom son: 1) conocimiento, 2) comprensión, 3) aplicación y 4) análisis. Donde el de mayor jerarquía implica a los anteriores, siendo una taxonomía incluida.

Abstract "Intelligent Learning Systems with Learning Objects (ProgEst)"

Learning environments are changing. The new scenarios pose technical and pedagogical challenges to which higher education institutions must respond. Roles for professors, students, and support personnel need to adapt to new contexts. For the case of this work, the main objective is to combine new technologies with artificial intelligence.

The following work describes the analysis and design of an ILS through structured programming. The organization of such ILS is based in LO.

An ILS is developed, basing the teaching-learning process on an inference engine, inspired by the human tutorial process, and is comprised by nine elements: 1) interest, 2) desire, 3) help, 4) cognitive and operational strategies, 5) interruption, 6) resignation, 7) learning, 8) inactive times, 9) error. Such elements are related in a causal matrix that allows causality to be observed between each one of them with the others. In addition to these nine elements, two more are added that allow for the enrichment of the student model, considering the learning style that will have a direct impact on the interface, and the internal motivation of the study that will permit choosing operational strategies in accordance with the latter. The inference engine is represented through diffuse cognitive maps. The latter is not a part of the development of this work.

On the other hand, the curriculum is designed using a genetic graph, which is based on a multi-agent architecture, implying three expert agents, each one in a sub-domain. The sub-domains are represented by instructional objects. The above allows for the elaboration of a teaching-learning process based on scenarios, linked to each sub-domain, and thus allowing for a detailed error management.

The instructional objectives involved in this work, according to Bloom's taxonomy are: 1) knowledge, 2) comprehension, 3) application, and 4) analysis. The one of greater hierarchy includes the others, being an included taxonomy.

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1 "Introducción"

En este capítulo abordaremos la definición según citan distintos autores de las nuevas tecnologías y su aplicación en la educación, su evolución y su conexión con los objetos de aprendizaje y los sistemas de aprendizaje inteligentes.

También en este capítulo se describen los objetivos generales y particulares del trabajo de tesis. Y además se incluye una breve descripción del trabajo.

1.1 Nuevas Tecnologías

Los principales problemas de la educación actual son: abundancia de conocimientos, anacronismo e inadecuación. Destaca la falta de calidad de la enseñanza actual y una desilusión creciente ante el aula tradicional, debido a que la forma de enseñanza continúa desfasada de las necesidades y de los medios existentes en el mundo actual, esto se debe a que la didáctica va detrás del desarrollo técnico.

A esto es necesario añadir algunas características de los años 90, como la internacionalización de la economía, la globalización de la comunicación e información y el empleo diversificado. Al respecto, señalan (Marchesi y Martín, 1998), que los sistemas educativos deben preparar a sus alumnos para las nuevas demandas. Los cambios educativos son inevitables y necesarios. "Estando inmerso el sistema educativo en una sociedad en constante transformación, no es posible pensar que la institución educativa puede mantenerse alejada de las modificaciones que permanentemente van sucediendo".

Entre las razones para el deterioro de la enseñanza cita (Pazos, 2001), la baja calidad de la enseñanza, la desilusión generalizada y creciente por el aula tradicional y el mal uso de los medios actuales, cuyas deficiencias deberán conocerse.

(Salinas, 1995), destaca la necesidad de nuevos estilos de enseñanza que conduzcan a adecuar a los tiempos de cambio a los futuros profesionales. Esto

supone lograr una enseñanza más activa, así como un mayor protagonismo de los estudiantes en su propio aprendizaje. (Ketudat, 2000), indica que se debe dar a las personas no sólo habilidades generales y vocacionales sino habilidades de aprendizaje y mentes perceptivas, el amor por aprender y habilidades de “aprender a aprender”, es decir de auto-aprendizaje. De Antonio, Barreiro, Crespo, Pazos, Rodríguez Patón y Silva (2003), y Lucas Marín, (2000), proponen que la educación sea universal, (esto es, que afecte a todos) y obligatoria: durante toda la vida, con procedimientos adaptables y sistemas educativos flexibles. Mayorga, (1999), refiere que en la educación del futuro será más importante aprender a aprender que memorizar contenidos específicos, la búsqueda y el uso de la información para resolver problemas que la transmisión de datos, los métodos activos y personalizados que los pasivos y estandarizados. También cambia la concepción de lo que significa ser maestro: transmisor de conocimientos a facilitador del proceso de aprendizaje que aprende continuamente él mismo.

Azpiazu, Pazos y Silva, (2001), proponen la teleformación y el aula virtual. La teleformación o tele-enseñanza permite la flexibilidad de tiempo y espacio y fomenta el auto-aprendizaje. Sin embargo, para que la práctica sea exitosa, se necesita un diseño didáctico y pedagógico riguroso, un uso adecuado de los recursos tecnológicos, incluyendo la creación de memorias institucionales para responder automáticamente consultas y una asistencia suficiente y personalizada. Como desventajas se pueden destacar la escasez de interacción entre los alumnos y el profesor. El aula virtual debe proporcionar una educación con centro en la resolución de problemas y búsqueda de soluciones. El profesor actuará como guía o timonel.

Marina, Feixas y Marqués, (1999), proponen considerar nuevos contenidos y competencias en el currículum, nuevos instrumentos y recursos para la docencia y su gestión, acceso a todo tipo de información, nuevos canales comunicativos para el aprendizaje y la colaboración, nuevos escenarios educativos asíncronos (flexibles, interactivos, personalizados) y nuevos métodos pedagógicos (nuevas formas de comunicación y aprendizaje, enseñar a aprender).

Una primera aproximación a la definición de entorno de aprendizaje, se tiene al indicar qué debería contener éste. El entorno de aprendizaje debe contener los elementos con los cuales el alumno construye sus modelos mentales, o sea debe dar las condiciones con las cuales cualquier alumno más o menos normal, estaría en disposición de aprender por si mismo. Ese entorno debe ser eficiente (que le permita asimilar esos conceptos y no se creen modelos falsos) y efectivo (esto es que facilite los modelos correctos). Asimismo debería fomentar la interactividad, quizá deban tener menos contenido pero contenido de mayor calidad. Para Wilson, (1995), un entorno de aprendizaje debe

contener como mínimo al aprendiz y un espacio donde el aprendiz actúa usando herramientas y dispositivos coleccionando e interpretando información, interactuando con otros, etc.

Los entornos de aprendizaje están cambiando. Los nuevos escenarios plantean desafíos técnicos y pedagógicos a los que las instituciones de educación superior deben responder. Los roles de los profesores, alumnos y personal de apoyo deben adaptarse *a los nuevos entornos, como lo sugiere Islas, (2001).*

Wilson, (1995) citado por Perkins en De Kereki, (2003), señala que los componentes principales de un entorno son:

Bancos de información: repositorios de información, ejemplos: libros, videos, maestros, etc.

"*Símbolo pads*": superficies para la construcción y manipulación de símbolos y lenguaje, ejemplos: cuadernos, tarjetas, procesadores de texto, programas de dibujo y de bases de datos.

"*Fenomenaria*", "áreas para presentación, observación y manipulación de fenómenos, ejemplos: acuarios, "*Sim City*".

Kits de construcción: similar a "fenomenaria", pero no tienen contraparte en el mundo real, ejemplos: "*Lego*", software de uso matemático y,

Manejadores de tarea: realiza las funciones de control y supervisión, indica las tareas, da apoyo, retroalimentación. Muchas veces esta tarea la realiza el profesor.

En JISC, (2000), se señala que los componentes principales de un entorno de aprendizaje virtual son:

"Mapeo" del currículum en elementos que puedan ser validados y registrados.

Registro de la actividad del estudiante y logros.

Soporte en línea incluyendo acceso a recursos de aprendizaje, evaluación y guía.

Soporte de tutoría y de pares.

Comunicación, incluyendo correo electrónico, discusión de grupos, acceso a Web y;

Vínculos a otros sistemas.

Otros elementos a tener en cuenta son:

- Establecer claramente los objetivos para el uso del entorno.
- Recursos necesarios para implementar y soportar el sistema.
- Capacitación necesaria.
- Costos del sistema.

Goñi, (2000), señala que se debe considerar la socialización del aprendizaje virtual, teniendo presente el ámbito de variedad de relaciones que se puede establecer. Debe tenerse también un sistema de información sobre la planificación de tareas, para asignar recursos, medir esfuerzos y dedicaciones estableciendo indicadores de progreso, un sistema de estándares y de valoración de las unidades didácticas, y los aspectos ya citados de un sistema de apoyo (ayudas, guías, consultas) y un sistema de evaluación.

Según Miklos y Arroyo, (2008), el nuevo modelo de sociedad que están lentamente perfilando las nuevas tecnologías, ha sido definido con una amplia variación de nombres: como infocapitalismo, capitalismo tardío, sociedad tecnotrónica, postcapitalismo industrial, sociedad de la información, etc. Pero todas las diversas definiciones se pueden reducir a una sociedad global que expande su economía a través de la innovación, el saber y la sustitución del empleo de baja capacitación por procesos automáticos expandidos gracias a trabajadores simbólicos. También todas las conceptualizaciones se orientan a definir el conocimiento como el factor clave y motor central de la nueva sociedad. Es así que se formulan las definiciones de capital humano, capital intelectual, capital ingenieril, capital cultural, de empresa en red, o simplemente de innovación. Todas ellas remiten en diversas formas, al proceso mundial en el cual se está creando una nueva sociedad global, como una división internacional del trabajo asociada a tijeras de precios derivadas de la densidad tecnológica de la producción, y donde el conocimiento, en proceso de mercantilización, crecientemente se localiza en los países centrales que se redefinen como los centros de producción, apropiación y utilización intensiva de los nuevos saberes protegidos por el derecho intelectual.

El actual escenario de revolución de los saberes y las tecnologías de información y comunicación, están conformando toda la educación y no sólo la educación a distancia hacia una nueva educación virtual global en red. Así, la revolución, de la digitalización, la microelectrónica barata y las telecomunicaciones, promueven la conformación de un nuevo paradigma educativo, una nueva organización institucional, disciplinaria, geográfica, económica y pedagógica de las instituciones educativas marcado por la

despresencialización a través de nuevas modalidades de comunicación en mercados globales y competitivos.

Según (Miklos T. et al., 2008), las mega universidades globales y las redes digitales colaborativas de comunicación, la educación transdisciplinaria, flexible y de movilidad a escala internacional, los modelos de simulación y los hipertextos no lineales de aprendizaje que implican escalas masivas de producción, así como la propia alta diferenciación y renovación disciplinaria y las demandas de una educación permanente, son apenas algunas derivaciones de los impactos sobre los procesos educativos que marcan un cambio planetario en tanto crecientemente son las tecnologías de comunicación e información las que sientan las bases del proceso de enseñanza-aprendizaje, en tanto viabilizan un amplio conjunto de cambios organizativos, conceptuales, geográficos, epistemológicos y pedagógicos en la educación y que en los tiempos presentes asumen la forma de una educación en red, tanto sea virtual, a distancia o presencial.

Los Sistemas y Modelos Educativos, como toda la estructura de la sociedad, requieren acciones encaminadas hacia la innovación y la adecuación que demandan integrarse en la sociedad de la información y el conocimiento, y mantener el equilibrio en medio del vertiginoso ritmo que el acceso de la tecnología imprime al entorno mundial. En este contexto, las instituciones y los procesos que tienen lugar en su interior, deben incluir en sus objetivos, lo que sólo hasta hace unos años se percibió como una prioridad para asegurar el avance de los ciudadanos y de la sociedad: la formación permanente de los ciudadanos a través de ofertas innovadoras, accesibles, pertinentes y satisfactorias en sus contenidos y metodología para las personas.

En este punto, adquiere relevancia el uso de la tecnología en proyectos educativos diversos, siempre relacionados con objetivos superiores y con la calidad pedagógica que demanda la globalización de la economía y en general la cultura. Bajo las circunstancias actuales se requiere revisar técnicas y estrategias que permitan formular propuestas que si bien atiendan la problemática en lo inmediato, su vigencia sea de largo alcance. **Las tecnologías digitales en tanto cambian las estructuras de costos, viabilizan nuevas modalidades de cobertura, promueven nuevas modalidades de aprendizaje, transforman la centralidad del aula y el rol de los docentes, y sientan las bases de un nuevo modelo educativo global desnacionalizado y despresencializado. La complejidad del proceso sin lugar a dudas genera tensiones, por ende diversos tipos de resistencias a su generalización como modelo educativo.**

Según Rama, (2006), en América Latina la educación a distancia se ha manifestado desde los inicios de la incorporación de las nuevas tecnologías.

Para el año 2000 ya existían 134,527 alumnos de modalidades virtuales, híbridas o a distancia, en 175 Universidades en su mayoría públicas, cobertura que para entonces representaba el 1.3% del total de estudiantes de la región. Sin embargo los estudiantes de modalidades híbridas eran dominantes, para el 2003 otro estudio determinó la existencia de 459 instituciones de educación superior de la región que estarían ofreciendo algún tipo de programas de educación virtual, pero de ellas, sólo 149 instituciones estarían ofreciendo programas virtuales en sentido más estricto, en tanto que 310 instituciones ofertarían bajo modelos fundamentalmente híbridos. El total de instituciones en ambas modalidades, representaban el 12.7% de las instituciones de los países reportados en los mencionados estudios nacionales y el 5.5% del total de instituciones existentes en la región. Recientemente otro relevamiento en el 2005 en el marco de la UNED de España determinó que la presencia mayoritaria es ahora de la educación virtual privada y también de modalidades híbridas.

1.2 Definición de Tecnologías de la Información y Educación a Distancia

Según Miklos, et al., (2008), algunos autores definen a las tecnologías de la información (TI) como la serie de metodologías, herramientas, técnicas y dispositivos utilizados en el manejo y proceso de la información, dentro del ámbito de la informática y la computación.

El campo de la educación a distancia no es un invento de fin de siglo. Esta modalidad educativa tiene antecedentes ubicados en diversas experiencias que en el mundo se han dado desde épocas remotas. Se pueden rastrear ejemplos diversos en los que pueblos o sujetos que se impusieron la tarea de educar, y cuyos educandos se encontraban en locaciones dispersas, tuvieron que desarrollar estrategias para hacer llegar sus mensajes utilizando los medios de comunicación a su alcance.

La educación a distancia, como parte de la educación formal o instituida, se desarrolla en el siglo XX en todo el mundo como modalidad alternativa asociada al concepto de oportunidad para poblaciones tradicionalmente alejadas de los sistemas escolares. No es casual que los países pioneros que han alcanzado mayor desarrollo en esta modalidad, sean aquellos con grandes extensiones territoriales, como Australia, Canadá, India, entre otros, y los que tuvieron necesidades de expandir política, que formaron social y culturalmente un imperio, como el caso de Inglaterra y la influencia que ejerció en los dos países arriba mencionados. En América Latina, países como Venezuela, Brasil, Colombia y Costa Rica han alcanzado un importante desarrollo en estas

modalidades. Importante no sólo por su consolidación sino por la diversidad de necesidades educativas que han reconocido en su población y la apuesta a la educación como factor de desarrollo social.

Las razones históricas para la expansión y consolidación de estas modalidades son sumamente interesantes para entender las particulares formas en que las políticas educativas de las naciones se orientan a la no convencionalidad educativa y las estrategias que han instrumentado para hacerlas posibles.

Según Miklos, et al., (2008), educación a distancia (ED) significa distribución de educación que no obliga a los estudiantes a estar físicamente presentes en el mismo lugar donde se encuentra el instructor. En un nivel básico, la ED se realiza cuando los estudiantes y maestros están separados por la distancia física y la tecnología (voz, video, datos e impresiones) a menudo en combinación con clases cara a cara, es utilizada como puente para reducir esta barrera (*Distance Education at a Glance*).

En la actualidad se utilizan una gran variedad de medios electrónicos para enviar o recibir los materiales de apoyo teniendo como objetivo la ED. Cada institución determina los medios más convenientes, dentro de los que tiene a su alcance y sus alumnos también, y con ellos realiza las combinaciones que mejor se adapten a sus posibilidades.

Desde la entrada de Internet en nuestras vidas, las posibilidades de acceso a la formación se han ido incrementando en la medida en que la red informática nos va permitiendo acceder a más personas y ofrecer ambientes de aprendizaje más complejos y elaborados.

El *e-learning*, otra de las modalidades del aprendizaje a distancia tiene, entre sus principales ventajas, la facilidad de acceso. La formación puede llegar a más personas, puesto que desaparecen las barreras espacio-temporales. De esta forma, personas que antes tenían dificultades para estar en contacto continuo con los procesos de formación, por problemas de desplazamiento al centro donde se imparten los cursos, por escasez de tiempo, por incapacidad física para asistir a clase, por vivir en pequeñas aldeas poco comunicadas con el exterior, etc. tienen ahora todo un abanico de posibilidades a su disposición para una formación continua.

Actualmente con el avance de las tecnologías de la información y las comunicaciones se han incrementado las posibilidades de desarrollo de modalidades educativas alternativas, que de alguna manera permiten llegar a más demandantes con mayor efectividad en el proceso de enseñanza-aprendizaje. Se utilizan una gran variedad de medios electrónicos para enviar o recibir los materiales de apoyo teniendo como objetivo la educación a distancia. Es ahí donde nace la modalidad *e-learning*. Este término se refiere a la

utilización de nuevas tecnologías de la información y la comunicación con un propósito de aprendizaje. Una de estas tecnologías es internet, pero también se pueden incluir los conceptos de multimedia y los simuladores. Como lo menciona Casas Armengol y Pomerhanz Stojanovic, (2008), se puede **decir que el *e-learning,* es la modalidad de educación a distancia, donde tanto el profesor como el alumno hacen uso de medios electrónicos para llevar a cabo el proceso de enseñanza-aprendizaje.** También podríamos hablar de *educación digital* pero el *e-learning* no es un entorno absolutamente digital, por lo que parece más adecuada la denominación de *e-learning* o *aprendizaje electrónico*, o bien *enseñanza electrónica* o de *educación electrónica*.

Definimos interoperabilidad como la capacidad de aprovechar información producida en sistemas heterogéneos. Lo anterior integrado en el diseño de un sistema, que en nuestro caso son Sistemas de Aprendizaje Inteligentes (SAI). La interoperabilidad implica el uso de *e-learning*, además de una plataforma tecnológica. Con el fin de lograr lo anterior hay que reflexionar acerca de otros elementos como son: 1) el modelo pedagógico que se usa en los materiales didácticos producidos, 2) la población para la cual se está creando el material, y 3) el uso de Objetos de Aprendizaje (OA), entre otras cosas.

El Modelo de Objetos de Aprendizaje (OA) ofrece una manera de construir contenidos educativos por composición a partir de piezas de elementos que se ubican en los niveles inferiores. Así mismo como una forma de buscar objetos y contenidos, localizarlos, recuperarlos e integrarlos a través de una colección de especificaciones y estándares para e-learning basado en la Web, denominada SCORM (Shareable Content Object Reference Model) para su catalogación, requisición, exportación, transportación e importación. Finalmente ofrece la oportunidad de construir, para cada estudiante y cada momento, una selección personalizada de contenido educativo que le brinde un contexto óptimo para su aprendizaje.

La anatomía de un OA, se observa la Figura 1.1. (Laureano-Cruces, Sánchez-Guerrero, Ramírez-Rodríguez & Mora-Torres, 2008a).

Figura 1.1 Componentes del Metadato.

Estos nuevos medios están siendo utilizados para expresar conocimiento, presentar información y guiar actividades de aprendizaje en los materiales. El incremento en la riqueza de los contenidos representa sin duda un cambio en el contexto educativo, pero no deja de ser un cambio menor si no es acompañado de trasformaciones más profundas como: 1) la forma de organizar los contenidos educativos, 2) el modo de acceder a ellos, y 3) su uso en el proceso de enseñanza aprendizaje.

Por otro lado los Sistemas de Aprendizaje Inteligentes (SAI) son interesantes debido a que simulan y hacen explícita una clase importante de inteligencia humana; y es la enseñanza en el nivel de abstracción más alto. Los Sistemas de Aprendizaje Inteligentes (SAI) pueden ser vistos como agentes activos que adaptan sus estrategias de enseñanza basándose en los cambios que percibe del estudiante, cuando este último se encuentra en un proceso de aprendizaje (Laureano-Cruces, 2000; Laureano-Cruces, Terán-Gilmore & De Arriaga, 2004a; Laureano-Cruces, De Arriaga & El Alami, 2004b; Laureano-Cruces, Ramírez-Rodríguez, De Arriaga & Escarela-Pérez, 2006 y Laureano-Cruces, Ramírez-Rodríguez, Mora-Torres, De Arriaga & Escarela-Pérez, 2008b). La descripción de las estrategias de enseñanza, así como el control, son aspectos que tienen un papel importante en la construcción de SAI.

Los Sistemas de Aprendizaje Inteligentes (SAI) están integrados por cuatro componentes: 1) el módulo experto, 2) el modelo de estudiante, 3) la interfaz, y 4) el módulo tutor.

En los SAI el término de inteligencia se asocia a la capacidad de adaptación dinámica a diferentes tipos de estudiantes. Lo anterior lo logran incorporando las elementos antes mencionados en: 1) el conocimiento que el sistema tiene del dominio, 2) los principios del proceso tutorial y los métodos bajo los cuales son aplicados, y 3) la representación del conocimiento que supuestamente tiene el usuario.

Los SAI enfocan el proceso de aprendizaje como una cooperación entre el tutor y el alumno. El tutor basándose en la percepción del alumno decide en cada momento qué estrategia es adecuada. Estas estrategias serán elegidas en base a la medida de una serie de parámetros como son: los errores cometidos, el estilo de aprendizaje, los conocimientos dominados, etc. Todo lo anterior para poder decidir: qué explicar, con qué nivel de detalle, cuándo y cómo interrumpir al alumno (Laureano-Cruces, 2000; Laureano-Cruces, et al, 2004a y 2004b y Laureano-Cruces, et al, 2008b) .

1.3 Objetivos Generales y Particulares del trabajo de tesis

Los objetivos generales de este trabajo de Tesis son:

- Conjuntar las nuevas tecnologías con la inteligencia artificial. En el caso específico del trabajo: los Objetos de Aprendizaje (OA) con el Sistema de Aprendizaje Inteligente (SAI).
- Diseñar un Sistema de Aprendizaje Inteligente.
- Flexibilizar el aprendizaje con base en las necesidades del alumno aprovechando las nuevas tecnologías (Información y comunicación), y las técnicas de inteligencia artificial aplicadas a la educación.

Los objetivos particulares son:

Contar con una estructura basada en un lenguaje de programación XML, y con un estándar internacional de interoperabilidad (SCORM para efectos del proyecto), que garantice la utilización en plataformas con distintos ambientes de programación.

Construir los objetos de aprendizaje de programación estructurada.

Desarrollar el SAI como herramienta que facilite el uso de entornos dedicados al proceso de enseñanza-aprendizaje integrando materiales didácticos y herramientas de comunicación y colaboración a través de la Web.

1.4 Propuesta

En esta tesis se desarrolla un SAI llamado Sistema *ProgEst cuyo objetivo es conjuntar* las ventajas que nos proporcionan: los Sistemas de Aprendizaje Inteligentes (SAI), las Tecnologías de la Información y la Comunicación (TIC), el *e-learning* y los Objetos de Aprendizaje (OA) (Laureano-Cruces, et al., 2008a) y (Sánchez-Guerrero L., Laureano-Cruces, Ramírez-Rodríguez & Mora-Torres, 2009). Lo anterior con el fin de utilizar los objetos de aprendizaje en ambiente de e-learning menos rígidos.

En este caso, se ha diseñado un SAI, para su implementación y dicha implementación está relacionada con los objetos de aprendizaje.

El SAI incluye técnicas de inteligencia artificial que apoyan la instrumentación del proceso de enseñanza-aprendizaje. El proceso está basado en un grafo genético, cuyos nodos en el nivel más alto de abstracción representan los objetivos instruccionales. Siguiendo la arquitectura propuesta por Laureano-Cruces, (2000a) y Laureano-Cruces A. & De Arriaga, (2000b) estos objetivos quedan inmersos en una arquitectura multiagente.

Para lograrlo se utilizó un motor de inferencia, basado en un *tutor de didáctica general* propuesto por Laureano-Cruces, et al., (2008b). Éste se liga al módulo tutor del caso de estudio y al comportamiento percibido por parte del usuario (modelo de estudiante). El comportamiento del usuario está representado por el modelo de estudiante. A través de la relación entre el módulo tutor y el modelo de estudiante se establece una relación que permite elaborar las diferentes estrategias didácticas.

El modelo de didáctica general que se utiliza en esta implementación considera los nueve elementos propuestos por Laureano-Cruces, et al., (2000b). Dicho modelo está inspirado en el comportamiento humano. Estos elementos están relacionados a través de causalidades que les permiten considerar la influencia de cada uno sobre los demás. La forma de representación elegida para darle vida a este motor de inferencia, es una técnica conocida como Mapas Cognoscitivos Difusos (MCD) y pertenece al área de ingeniería cognoscitiva y métodos heurísticos.

Los SAI plantean el proceso de aprendizaje como la cooperación entre un sistema inteligente y el ser humano. El tutor, basándose en la evaluación del desempeño del usuario, se encuentra en un *proceso constante de toma de decisiones*, con el fin de elegir la estrategia más apropiada de enseñanza. Estas estrategias serán elaboradas con base en la percepción del desempeño del usuario tomando como evidencia una serie de parámetros como pueden ser: errores cometidos, estilo de aprendizaje, dominio de los conocimientos, y estado afectivo-motivacionales, entre otros. Estas valoraciones sirven como referencia para determinar: qué explicar, a qué nivel de detalle, y en qué momento, así como: cuándo interrumpir al alumno, y qué información proporcionar durante la interacción (Laureano-Cruces, 2000; Laureano-Cruces, et al, 2004a y 2004b y Laureano-Cruces, et al, 2008b).

1.5 Estructura del Trabajo de Tesis

Esta memoria está organizada de la siguiente forma:

En el ***capítulo dos*** se describen los sistemas de aprendizaje inteligente, haciendo énfasis en una arquitectura multiagente.

En el ***capítulo tres*** se describe brevemente el tema de las nuevas tecnologías, aplicadas al aprendizaje a distancia (e-learning), así como la anatomía de un objeto de aprendizaje y la metodología utilizada para el diseño de éstos.

En el ***capítulo cuatro*** se describe el dominio de programación estructurada en seudocódigo, tomando como base el grafo genético y los objetivos instrucionales (éstos serán representados por la arquitectura multiagente).

En el ***capítulo cinco*** se presenta el diseño del Sistema de Aprendizaje Inteligente ProgEst. Se describe el diseño del: 1) módulo tutor, 2) modelo del estudiante, y 3) la interfaz. A través de escenarios específicos se describen la interacción del proceso de enseñanza-aprendizaje, considerando los errores y su clasificación.

En el ***capítulo seis*** se detalla el diseño de los objetos de aprendizaje del Dominio de Conocimiento "Programación Estructurada con Algoritmos en pseudocódigo", aplicando la metodología seleccionada para su construcción.

En el ***capítulo siete*** se detalla el funcionamiento del sistema ProgEst con ejemplos prácticos.

En el ***capítulo ocho*** se describe la arquitectura por componente del sistema ProgEst.

En el ***capítulo nueve*** se presentan las conclusiones y las propuestas de trabajos futuros.

Capítulo 2 “Sistema de Aprendizaje Inteligente (SAI)”

2.1 Sistemas de Aprendizaje Inteligentes

Actualmente el avance acelerado de la tecnología en el incremento de la capacidad de procesamiento de los sistemas de cómputo nos permite la inclusión de técnicas de Inteligencia Artificial en cuanto al manejo y elaboración de los Sistemas de Aprendizaje Inteligentes (SAI) y enfocarnos a la nueva área de desarrollo de e-learning. (De Arriaga, El Alami, Laureano-Cruces, Martínez y Arriaga-Perelló, 2002)

Como lo describe Laureano-Cruces, (2000a) y Laureano-Cruces, et al., (2000b), los primeros sistemas de aprendizaje que hicieron su aparición fueron los llamados Computer Aided Instruction (CAI). Estos últimos han evolucionado desde sus inicios a la fecha; utilizando diferentes técnicas que faciliten su uso y los vuelvan más atractivos.

La incorporación de las técnicas de inteligencia artificial (a partir de los 70) a estos sistemas dieron origen a los sistemas de aprendizaje inteligente. El término de inteligencia se asocia a la capacidad de adaptación dinámica a diferentes tipos de estudiantes. Lo anterior lo logran incorporando las técnicas mencionadas en: 1) el conocimiento que el sistema tiene del dominio, 2) los principios del proceso tutorial y los métodos bajo los cuales son aplicados, y 3) la representación del conocimiento que supuestamente tiene el usuario.

Los SAI enfocan el proceso de aprendizaje como una cooperación entre el tutor y el alumno. El tutor basándose en la percepción del alumno decide en cada momento qué estrategia es adecuada. Estas estrategias serán elegidas en base a la medida de una serie de parámetros como: errores cometidos, estilo de aprendizaje, conocimientos dominados, etc. Lo anterior para poder decidir: qué explicar, con qué nivel de detalle, cuándo y cómo interrumpir al alumno.

La investigación en el área de los SAI, ha estado centrada en diferentes aspectos, desde la inspección de cada uno de sus módulos constituyentes hasta la elaboración de arquitecturas genéricas (Arruarte, 1998). Así como el aprendizaje automático y la construcción de ayudas de diseño de sistemas de enseñanza.

El dominio del SAI se encuentra en la intersección de tres diferentes áreas: 1) las ciencias de la computación (inteligencia artificial), 2) la pedagogía (recursos educacionales) y 3) la psicología cognitiva (métodos de análisis de los diferentes procesos cognitivos), ver Figura 2.1. Además se encuentra el área del conocimiento que se pretenda enseñar a través del sistema.

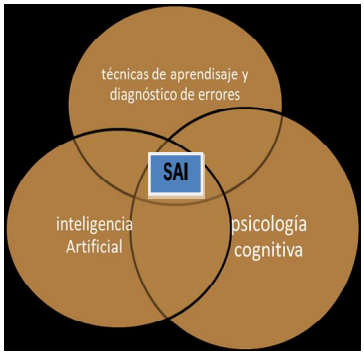


Figura 2.1 Dominio de un SAI.

Los sistemas de aprendizaje inteligentes (SAI) están integrados por cuatro componentes: 1) el módulo experto, 2) el modelo de estudiante, 3) la interfaz, y 4) el tutor (ver Figura 2.2). A continuación se da una breve descripción. Para mayor detalle consultar Laureano-Cruces, et al., (2000b) y Laureano-Cruces, (2000a).

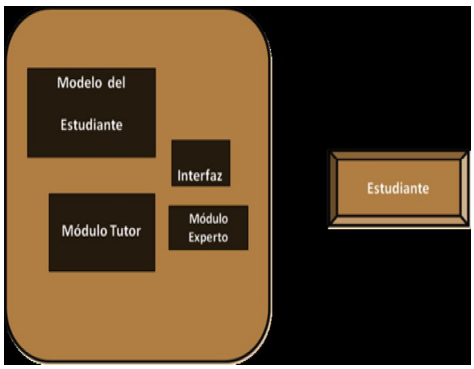


Figura 2.2 Componentes Básicos de un SAI.

El módulo experto.- es el lugar donde se encuentra acumulado el conocimiento que el sistema intenta enseñar al estudiante. La implementación de este componente está íntimamente ligada al módulo tutor. Debido a que el tutor enseñará el dominio haciendo énfasis en la organización del módulo experto. De aquí que interese que este módulo esté organizado de forma pedagógica.

El modelo de estudiante.- es una base de datos que contiene información del estudiante que permite desarrollar las siguientes funciones: 1) adaptación del sistema con base en la competencia que tenga el estudiante de un determinado material (objeto de la enseñanza), 2) hacer un reporte del material cubierto de acuerdo al currículo, 3) seleccionar el nivel adecuado de intervención y explicación, 4) dar alguna ayuda de tipo operativo y 5) facilitar la retroalimentación del estudiante.

El módulo tutor.- tiene la responsabilidad de decidir qué acciones tomar para enseñar o corregir un determinado dominio basándose en el diseño del currículo. Es él quien selecciona los problemas para ser resueltos por el estudiante, analiza las respuestas, presenta la solución de ciertos problemas o decide mostrar algunos ejemplos. Lo anterior en relación con los objetivos que el planificador[1] tiene; con respecto a uno o varios temas específicos a enseñar.

Gestiona el material didáctico y se encarga de seleccionar el material más adecuado en función de las situaciones reportadas. Estas situaciones son principalmente determinadas por las demandas del planificador y del comportamiento del alumno percibido a través de la interfaz.

La interfaz.- puede ser considerada como un entorno de simulación donde tienen lugar las salidas y entradas del sistema. Su responsabilidad básica es la comunicación entre el sistema y el estudiante, aunque al ser el medio de salida de las acciones del SAI, también tiene una responsabilidad didáctica (Velasco-Santos, Laureano-Cruces, Mora-Torres y Sánchez-Guerrero, 2008).

[1] Se entiende por planificador el lugar donde se encuentran implementados los métodos de control de la selección y secuenciación de las estrategias de enseñanza. Puede ser implementado como un componente aparte o como parte del módulo tutor. Dependiendo del mecanismo de control elegido.

2.2 Interacción Dinámica en un Sistema Multiagente

2. 2.1 La Inteligencia Artificial Distribuida

La inteligencia artificial distribuida (IAD) descentraliza el proceso de control; lo que lleva al uso de agentes autónomos. Estos agentes son especialistas que cooperan entre sí, de forma contraria a como lo hacen los sistemas que cuentan con un control general. Con lo anterior se trata de minimizar en lo posible la incapacidad de planificar dentro de un mundo dinámico, al permitir la existencia de varios especialistas resolviendo un mismo problema en lugar de uno. (Girard, Gauthier & Levesque, 1992)

2.2.2 Los Agentes Reactivos

Los agentes reactivos dotan a los sistemas de la capacidad de reaccionar con rapidez en un mundo dinámico y en continuo cambio. Para lograrlo cada agente sintetiza los aspectos de ese mundo que son relevantes para su funcionamiento. Con esta división se creará por otro lado, la interacción de estos agentes con el mundo y entre ellos para que el funcionamiento global del sistema emerja. Lo anterior de acuerdo a la tarea específica para la que fue creado el sistema.

Como consecuencia de lo anterior y partiendo de la idea: que un sistema de enseñanza inteligente es un sistema que opera en un ambiente: cognitivo, impredecible, complejo y por lo tanto difícil de modelar. Laureano-Cruces, (2000a) y Laureano-Cruces, et al., (2000b), proponen un SAI con características reactivas.

La arquitectura incluye agentes reactivos que representan la experticia de cada una de las sub-habilidades necesarias para aprender la programación estructurada. (Ver Figura 2.3)

En este tipo de arquitectura el comportamiento de los agentes está dividido en dos subagentes; uno que realiza el diagnóstico, y su objetivo es observar el entorno (que consiste en el desarrollo del estudiante). A partir de esta observación se obtiene evidencia en base a las capacidades perceptuales del agente. Para saber si el estudiante usa, o no usa, o usa de forma incorrecta, la habilidad monitorizada y controlada de forma exclusiva por ese agente. De sus observaciones se detecta el o los errores cometidos y entonces entra en acción el otro subagente, representado por un MicroMundo cuya tarea consiste en crear un entorno que ayude al estudiante a aclarar sus dudas, todo mediante la

utilización de las tácticas didácticas que guiarán su intervención hasta que el alumno sea devuelto al entorno principal, donde se cometió el error.

La intervención del MicroMundo consiste en presentar una estrategia instruccional, conformada por una estrategia cognitiva y una operacional en este caso enfocada a la parte de la motivación de estudio. Es necesario mencionar que debido al enfoque reactivo, estos sub-tutores cuentan con la capacidad en todo momento de mostrar el uso de la tarea por un experto.

En el nivel de abstracción más general un agente de ProgEst está representado por la Figura 2.3. En dicha figura, la abstracción está representada por la tabla de errores y las tácticas didácticas, por el otro lado la presentación está simbolizada por los medidores y actuadores que son todos los mecanismos de percepción a través de la interfaz y todos los mecanismos de intervención tutorial, tales como: animación, texto, sonido, etc.

Siguiendo la metodología propuesta por Laureano-Cruces, (2000a), ver Figura 2.3 y tomando como base el grafo genético del dominio de conocimiento de programación estructurada con pseudocódigo (Ver Capítulo 5, Sección 5.2.1) se subdividió la habilidad en: estructuras de control, variables, otros componentes, operadores y abstracciones. Para implementarlas en un sistema MultiAgente, donde los agentes son reactivos y cada uno de ellos cuenta con un agente de diagnóstico y otro encargado de la elaboración del proceso de enseñanza-aprendizaje y corresponde propiamente a la sub-habilidad, ver Figura 2.3. Lo anterior cuenta con tres tipos de ventaja desde el punto de vista de la enseñanza: a) con base en los agentes reactivos se puede implementar un mayor detalle en las técnicas y estrategias tutoriales; al poder crear subexpertos en las distintas subhabilidades, b) en un SAI reactivo quedaría implícito el orden de aprendizaje de una estrategia. El hecho de no utilizar alguna subhabilidad en un determinado momento; donde ésta sea necesaria dentro de ese contexto, provocaría la reacción inmediata del sistema y c) en el caso que se pretenda enseñar no una sino varias estrategias pertenecientes a un mismo domino, es muy probable que algunas de ellas compartan subhabilidades; con lo que se lograría una re-utilización de los agentes reactivos, así como del material utilizado por ellos y que en el caso de estudio se encuentra relacionados los objetos de aprendizaje y el material de apoyo al proceso de enseñanza-aprendizaje.

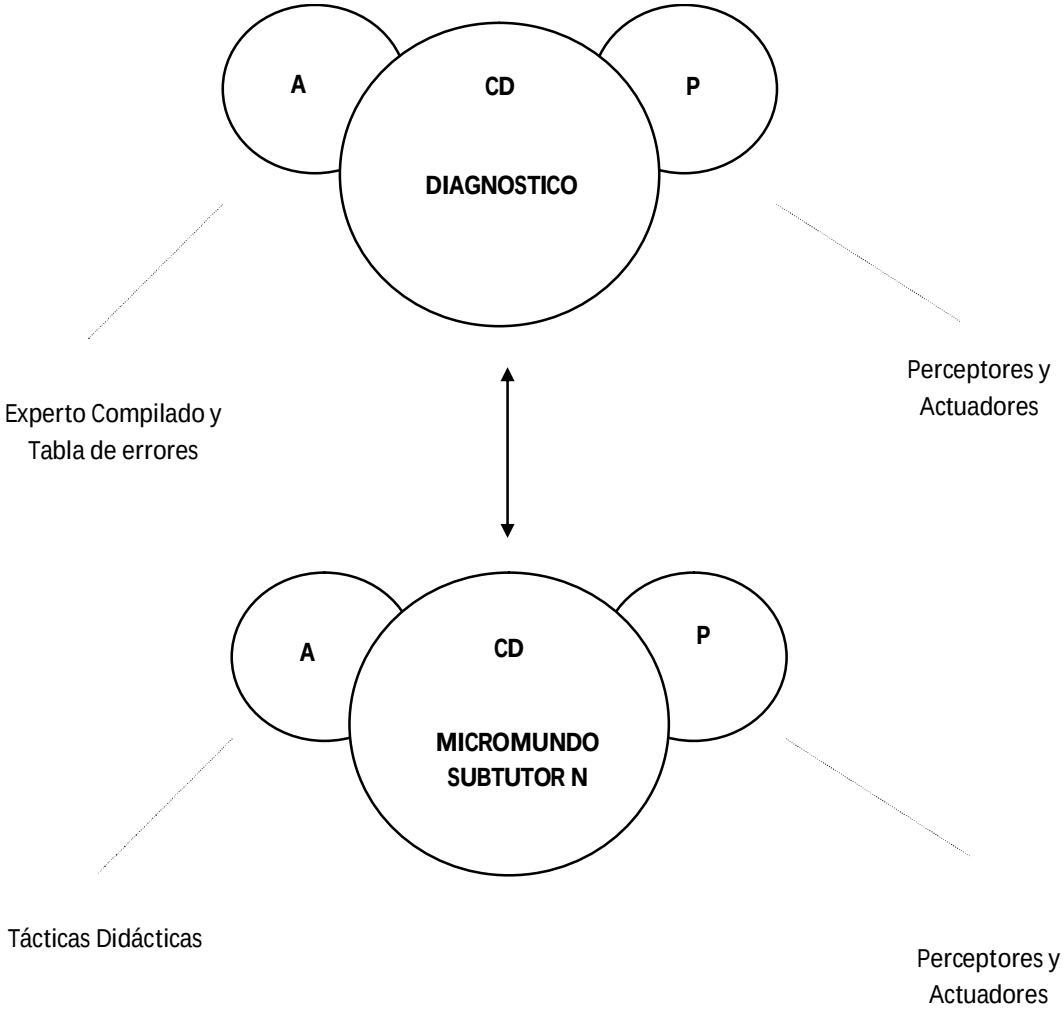


Figura 2.3 Agente Global del Sistema.

(Laureano Cruces A., 2000a)

En un sistema de enseñanza inteligente reactivo, el mecanismo de arbitraje está basado en prioridades. Estas últimas pueden establecerse por la importancia de los errores durante el proceso de diagnóstico de los mismos; dentro del contexto global de solución del problema, o por el orden de enseñanza de las sub-habilidades que involucran habilidades de mayor complejidad; en cuyo caso la jerarquía de la importancia de los errores quedará supeditada al orden de aprendizaje de las subhabilidades. En este caso se basa en errores (Laureano-Cruces, Ramírez-Rodríguez, De Arriaga, & Escarela-Pérez, (2006).

En este trabajo se diseña un SAI, con el fin de lograr su implementación. Para lograrlo se acopla un motor de inferencia, ligado al módulo tutor y al comportamiento percibido por parte del usuario. En este tipo de arquitectura el proceso de enseñanza-aprendizaje queda inmerso en la interacción dinámica

entre los agentes (sub-tutores) y el entorno (interfaz de desarrollo). Para lograrlo se utilizó un motor de inferencia basado en un modelo de didáctica general, el cual acoplado a nuestro sistema y relacionado con el domino específico; representado por el grafo genético de programación estructurada, genera las estrategias instruccionales específicas y ad-hoc.

El modelo de didáctica general que se utiliza en esta implementación considera los nueve elementos del tutor de didáctica general propuestos por: Laureano-Cruces, et al., (2008b), más el estilo de aprendizaje y la motivación de estudio (sea ésta interna o externa) conformados de la siguiente forma: 1) interés, 2) deseo, 3) ayuda, 4) estrategias cognoscitivas y operativas, 5)interrupción, 6) renuncia, 7) aprendizaje, 8) tiempos inactivos, 9)error, 10) estilo de aprendizaje, 11) motivación interna y externa. Dicho modelo está inspirado en un comportamiento humano. Estos elementos están relacionados a través de causalidades que les permiten considerar la influencia de cada uno sobre los demás. La forma de representación elegida para darle vida a este motor de inferencia, es una técnica conocida como mapas cognitivos difusos y pertenece al área de ingeniería cognitiva y métodos heurísticos. El desarrollo puntual de esta parte no se trata en este trabajo de tesis. Para más detalle revisar Laureano-Cruces, et al., (2008a), y Laureano-Cruces, et al., (2008b).

El nodo referente a incentivos, da pie a distintas formas en que podrá interactuar el sistema a través de la interfaz y se refiere a las distintas estrategias cognitivas y operativas que se pueden utilizar (se tratará en el capítulo 5). Esta acción considerará todos los datos del modelo del estudiante como son: errores, material cubierto, estilo de aprendizaje, objetivo interno de aprendizaje, y la motivación de estudio en Laureano-Cruces, Mora-Torres, Ramírez-Rodríguez & Gamboa-Rodríguez, (2009). En el caso de estudio nos basamos en la propuesta hecha por Laureano Cruces, et al., (2008a*),* la cual complementada con la clasificación de errores en sub-habilidades, nos proporcionará las estrategias instruccionales específicas y personalizadas.

En los siguientes capítulos se tratará con detalle el análisis y diseño de ProgEst.

Capítulo 3 "Objetos de Aprendizaje"

En este capítulo se abordarán algunos aspectos relacionados con la educación y las nuevas tecnologías. Así mismo dentro de éste se describirán los conceptos de objetos de aprendizaje (OA) y el repositorio de OA (ROA).

3.1 El e-learning como Metodología de Aprendizaje

El e-learning es un sistema de aprendizaje basado en el uso de internet y de uso creciente tanto en el ámbito educativo como en el de formación empresarial y profesional, este sistema (Escorcia Saldariaga, 2006):

- **Posibilita** la incorporación de conocimientos mediante el empleo de contenidos interactivos que involucran al alumno en el desarrollo del curso.
- **Permite** mediante servicios de internet, el trabajo y la interacción grupal, que son característicos de la educación presencial.
- **Flexibiliza** y facilita la organización de los cursos al reducir total o parcialmente la coordinación física de las actividades.
- **Facilita** el mantenimiento y actualización de contenidos y su distribución.

Finalmente permite realizar la capacitación laboral y profesional en el momento que se requiere, donde se necesita, reduciendo costos y en forma compatible con actividades u obligaciones laborales, sociales o familiares y capacitar a más alumnos en menos tiempo. Es un complemento eficaz de todas aquellas actividades que requieren la presencia física del alumno.

En los últimos años el uso de las plataformas de e-learning o LMS (Learning Management System) en las universidades ha ido en aumento, por lo anterior se han propuesto diseñar cursos a distancia con tecnologías educativas propias, cada institución cuenta con materiales en línea de distintas disciplinas,

así como materiales en línea de la misma disciplina en un mismo departamento académico.

De acuerdo a Miklos y Arroyo García, (2006), hoy en día con el avance de las tecnologías de la información y las comunicaciones, el número de modalidades educativas alternativas ha aumentado, estos avances han permitido llegar a más demandantes con mayor efectividad en el proceso de enseñanza-aprendizaje.

Los nuevos medios electrónicos de que se dispone, están siendo utilizados para expresar conocimiento, presentar información y guiar actividades de aprendizaje en los materiales. El incremento en la riqueza de los contenidos representa sin duda un cambio en el contexto educativo, pero no deja de ser un cambio menor si no es acompañado de trasformaciones más profundas como: 1) la forma de organizar los contenidos educativos, 2) el modo de acceder a ellos, y 3) su uso en el proceso de enseñanza-aprendizaje.

Los adelantos en las tecnologías de información y comunicación están logrando la promesa de la disposición global de información, que se realiza en primera instancia a través de la red internacional internet. La propuesta de objetos de aprendizaje (OA) de proveer información y actividades integradas en una entidad, busca el intercambio de información pero más bien enfocado a objetivos de aprendizaje.

El incremento de uso de las Tecnologías de la Información y Comunicaciones (TIC´s) en todo el mundo es una realidad que no puede evitarse. Los avances de la tecnología y sus aplicaciones educativas son sorprendentes. El diseño educativo organizado con Objetos de Aprendizaje (OA) ha tenido un impulso en los últimos años y se ha colocado como una de las principales tendencias en el campo de la educación aprovechando las TIC´s.

A partir del año 2000 ha crecido el número de tareas de investigación y generación de acervos de objetos, por lo que se espera que se cuente cada vez con mayor número de recursos digitales disponibles para todo tipo de prácticas educativas.

Es por lo anterior, que nuestro trabajo ProgEst aprovecha la aplicación del e-learning, los Objetos de Aprendizaje y los Sistemas de Aprendizaje Inteligente.

Después de establecer la relación entre la educación a distancia (ED), el e-learning y los OA, abordaremos los conceptos teóricos de OA y los Repositorios de Objetos de Aprendizaje (ROA), los estándares relacionados con ellos y la metodología de desarrollo de los OA.

3.2 Objetos de Aprendizaje

El propósito de esta sección es abordar la definición de los OA y revisar metodología propuesta por Muñoz-Arteaga, Osorio-Urrutia, Álvarez-Rodríguez, Cardona-Salas, (2006), para el diseño de los OA, que permita desarrollar paso a paso. Diseñar objetos de aprendizaje (OA) es considerada como una tarea difícil, por ser multidisciplinaria y porque es necesario considerar aspectos pedagógicos y tecnológicos.

Antes de iniciar con la metodología para la elaboración de OA, definiremos el concepto de Objetos de Aprendizaje en base a las experiencias de los expertos académicos de las distintas instituciones educativas que han dedicado sus investigaciones a los OA y ROA.

3.2.1 Los Objetos de Aprendizaje como un Concepto Puente

De acuerdo a Chan Núñez, et al., (2006), una de las razones por las que la noción de objeto de aprendizaje ha cobrado tanta fuerza en el campo de la educación apoyada por las TIC´s, es el hecho de que éste se pueda considerar como un Concepto Puente entre la educación, la comunicación, el diseño y las Ciencias de la Computación, por nombrar sólo algunos campos.

Es importante mencionar que el término *Objeto de Aprendizaje* tiene una fuerte relación histórica en las ciencias sociales, si se considera que viene de la categoría objeto (elemento visible en toda filosofía del conocimiento desde la antigüedad), y por otra parte, en las ciencias de la computación se constituye en un paradigma con la programación O*rientada a Objetos.*

Los seres humanos construimos el conocimiento con base en los objetos con los que nos relacionamos, los que modelamos y los que se convierten en constitutivos del entorno de vida.

La actividad de aprendizaje y la información necesaria para la ejecución se encuentran contenidas dentro del objeto. Esto posibilita la articulación de los objetos entre sí para configurar estructuras de aprendizaje crecientes.

La definición de objeto de aprendizaje más difundida hasta ahora, y al mismo tiempo, por su sencillez, más discutida y usada como base de nociones más elaboradas, es aquella que lo plantea *como cualquier recurso digital que puede ser rehusado como soporte para el aprendizaje.* (Wiley, 2000)

El grupo mexicano de trabajo de la Corporación Universitaria para el Desarrollo de Internet 2 (CUDI), definió al objeto de aprendizaje como:

“Un objeto de aprendizaje es una entidad informativa digital desarrollada para la generación de conocimiento, habilidades y actitudes requeridas en el desempeño de una tarea, que tiene sentido en función de las necesidades del sujeto que lo usa y que representa y corresponde a una realidad concreta susceptible de ser intervenida”.

Definieron como propiedades de los OA los siguientes:

- **Subjetividad.** Los objetos son polivalentes, pues la significación de sus potencialidades recae en los sujetos que los usan.
- **Realidad.** El objeto de aprendizaje es un puente con una realidad concreta.
- **Historicidad.** La pertinencia histórica de los objetos tiene que ver con su construcción y distribución en función de las condiciones reales de acceso y uso de los educandos a los que se pretende atender.
- **Complejidad.** Los objetos aunque tienen una delimitación que los convierte en unidades materiales, están ligados en múltiples formas con otros objetos posibles.
- **Comunicabilidad.** Los objetos de aprendizaje contienen información, y su capacidad de representación supone la integración de múltiples lenguajes.
- **Integralidad.** Tener unidades que al ser consultadas individualmente tengan ya una estructura y que nos lleven a un objetivo de aprendizaje específico.
- **Unidad coherente.** Objetos como pequeñas unidades de aprendizaje cuyos elementos tienen relación íntima con el objetivo que persiguen.
- **Unidades autocontenibles y versátiles.** Cada objeto puede ser tomado independientemente y que tenga elasticidad.
- **Reusabilidad.** Cada objeto puede ser usado en diferentes contextos y para diferentes objetivos.
- **Escalabilidad.** Los objetos pueden ser agrupados en una larga colección de contenidos para conformar la estructura de un curso.

- **Debe ser clasificable.** Cada objeto debe contar con ciertos elementos que permitan clasificarlo en un metadatos (descriptores), que tenga las propiedades de que puede ser encontrado fácilmente.
- **Relevante.** Que responda a una necesidad, sea pertinente y posea agenda de utilización.

Esta definición tiene sentido principalmente en el contexto de las instituciones de educación superior que siguen la tendencia del diseño curricular por competencias y de diversas organizaciones dedicadas a la capacitación profesional basada también en normas de competencia laboral.

Como lo mencionamos anteriormente existe una gran variedad de definiciones generales de Objetos de Aprendizaje. Para continuar con nuestro trabajo retomaremos la definición de L'Allier, (1997): "La mínima estructura independiente que contiene un objetivo, una actividad de aprendizaje y un mecanismo de evaluación". Con las siguientes características:

- Es reutilizable.
- Es auto contenible.
- Es independiente de plataforma.
- Es escalable.

Y está integrado por los siguientes componentes:

1) Objetivo de aprendizaje.
2) Contenido informativo.
3) Actividades de aprendizaje.
4) Evaluación.

Todo OA debe tener una etiqueta que lo identifica, a esta etiqueta le llamaremos metadato y contiene información de sí misma, esto facilita la búsqueda en un repositorio de OA.

Figura 3.1 Componentes de un Objeto de Aprendizaje.

La Figura 3.1 nos muestra la estructura general de un OA, donde se pueden identificar los elementos específicos y técnicos que lo componen.

Los OA actualmente tienen como sistemas formales de acceso dos instancias, los contenedores de OA, conocidos como "*repositorios digitales*" y/o "*los sistemas administradores de cursos*, CMS por sus siglas en inglés (Course Management System) también conocidos como LMS por sus siglas en inglés (Learning Management System). En ambos casos es necesario un estándar de intercambio de información.

El estándar de intercambio de información con mayor uso actualmente es SCORM y su proceso es conocido como "*empaquetado*" de contenidos. Actualmente existen herramientas que facilitan el empaquetado, como RELOAD. Tanto la herramienta como el estándar representan la instrumentación de los OA para alcanzar la compatibilidad de OA.

Finalmente, se describe el proceso completo de la siguiente manera: 1) identificar o crear recursos de calidad (OA), 2) especificar los derechos de autor, 3) aplicar estándares o empaquetado (SCORM), 4) almacenar los recursos en un repositorio para facilitar su consulta o crear cursos en línea por LMS. Para mayor información consultar Chan Núñez, et al., (2006).

3.2 Un Modelo de Objeto de Aprendizaje

De acuerdo al trabajo de Aguilar Cisneros, Muñoz Arteaga, Pomares Hernández, (2004), un modelo de Objeto de Aprendizaje debe especificar claramente sus elementos dada su posibilidad para integrar de una manera flexible diferentes puntos de vista pedagógicos sobre los contenidos de estudio y las actividades a realizar por el usuario del objeto.

En trabajos previos Aguilar Cisneros, et al., (2004), se describe un modelo para un objeto de aprendizaje, donde la plantilla se conforma por los siguientes campos:

1) **Teoría.-** Esta área contiene la información de los objetos de aprendizaje, la cual puede ser texto, imágenes, etc. Aquí los aprendices encontrarán los conceptos sobre el tema abordado en el objeto de aprendizaje, que les permitan obtener una experiencia abstracta.

2) **Experimentación.-** Esta área contiene animaciones, simulaciones que permitirán al alumno experimentar y reflexionar los conceptos abordados en el área de la teoría. Los alumnos obtienen una experiencia directa y concreta del tema abordado en el objeto de aprendizaje.

3) **Evaluación.-** Esta área permite evaluar el conocimiento adquirido por el alumno mediante sus experiencias abstractas y concretas. El alumno también puede calificar el grado de utilidad del objeto de aprendizaje.

4) **Información relacionada.-** Esta área muestra al aprendiz referencias a otros objetos de aprendizaje relacionados con el tema. A través de estas ligas se pueden explorar diferentes objetos de aprendizaje.

Los objetos de aprendizaje al igual que muchas otras tecnologías, tienen diversidad de formas de uso y operación, por lo que es necesario presentar una clasificación de los mismos.

3.3 Taxonomía de los Objetos de Aprendizaje

Muñoz Arteaga, Álvarez Rodríguez, Chan Núñez, (2007), proponen las siguientes clasificaciones para la taxonomía de OA. Una fue seleccionada por su tipología, considerada como vigente y la otra por la complejidad y futuro de los objetos de aprendizaje.

Los objetos de aprendizaje se han clasificado según los recursos, con fines completamente pedagógicos (ver Tabla 3.1), (ASTD y SmartForce, 2003).

Tabla 3.1 Clasificación de Objetos de Aprendizaje para su uso pedagógico.

Tipos de Objetos de Aprendizaje	Subcategorías
Objetos de Instrucción	**Lección** **Worksops** **Seminarios** **Artículos** **White papers** **Casos de estudio**
Objetos de Colaboración	**Ejercicios monitores** **Chats** **Fotos** **Reunión en Línea**
Objetos Prácticos	**Simulación de Juegos de Roles** **Simulación de Software** **Simulación de Hardware** **Simulación de Codificación** **Simulación Conceptual** **Simulación Modelo de Negocios** **Laboratorios en Línea** **Proyectos de Investigación**

Otro tipo de clasificación es el generado por el grupo nacional de objetos de aprendizaje (Muñoz Arteaga, et al., 2005), el cual enumera cuatro tipos de objetos.

1) **Objetos informativos.-** Es el tipo de objeto que contiene los elementos de conocimiento junto con su evaluación en sus aspectos básicos, y que incluso pueden llegar a resolver alguna competencia.

2) **Objetos generativos.-** Este tipo de objetos genera más objetos de aprendizaje a partir de plantillas pedagógicas y tecnológicas que resuelven competencias diversas y por lo tanto, facilitan la solución de problemas específicos curriculares.

3) **Objetos de simulación.-** Este tipo de objetos contienen la instrumentación de partes de simulación de diversos tipos (se pueden observar algunos de ellos en la clasificación de la Tabla 3.1).

4) **Objetos colaborativos.-** Permiten el aprendizaje grupal a través de los elementos dentro del mismo objeto (independiente del LMS).

3.4 Los Metadatos

Todo OA debe tener una etiqueta que lo identifique de los demás, a esta etiqueta se le llama metadato y contiene información acerca del mismo, con esto se facilita su búsqueda en un repositorio de OA. Considerando cada uno de los elementos del OA, se puede observar su estructura general en la (Figura 3.1).

De acuerdo a Chan Núñez, et al., (2006), los metadatos consisten en un conjunto de propiedades de un documento. Por definición, los metadatos son datos, además de datos sobre datos (Berners-Lee, 2000).

Los metadatos se basan fundamentalmente en el empleo del lenguaje XML(extensible Markup Languaje) desarrollado por el World Wide Web Consortium (W3C), que es una forma flexible de crear formatos con información, que al mismo tiempo integren el formato y los datos dentro del World Wide Web y la red. El XML al igual que el HTML (Hypertext Markup Languaje) contiene símbolos que describen los contenidos en un archivo o página. Algunos de los metadatos o descriptores son: título, idioma, palabra clave, autor, ciclo de vida, etc., así también otros agregados por el autor como el tamaño.

3.5 El Estándar SCORM

El *ADL SCORM*, fue formado en 1997, la iniciativa ADL (*Advanced Distributed Learning*), es un programa del Departamento de Defensa de los Estados Unidos y de la Oficina de Ciencia y Tecnología de la Casa Blanca para desarrollar principios y guías de trabajo necesarias para el desarrollo e implementación eficiente, efectiva y en gran escala, de formación educativa sobre nuevas tecnologías Web. Este organismo recogió lo mejor de las iniciativas anteriores, refundiéndolas y mejorándolas en un modelo propio: SCORM (*Shareable Content Object Reference Model*). Este modelo proporciona un marco de trabajo y una referencia de implementación detallada, que permite a los contenidos y a los sistemas, utilizarlo para comunicarse con otros sistemas, obteniendo así interoperabilidad, reutilización, durabilidad y adaptabilidad.

Para estandarizar la comunicación entre los LMS y OA existentes se creó un conjunto de especificaciones que vincula las fortalezas de las especificaciones de las organizaciones líderes en este ramo como el modelo SCORM ADL, (2005), es un conjunto de estándares y especificaciones para compartir, reutilizar, importar y exportar OA. Éste describe cómo las unidades de contenidos se relacionan unas con otras a diferentes niveles de granularidad,

cómo se comunican los contenidos con el LMS, define cómo empaquetar los contenidos para importarse y exportarse entre plataformas y describe las reglas que un LMS debe seguir a fin de presentar un aprendizaje específico. SCORM es expandible e incluye a trabajos del Institute for Electrical and Electronics Engineers Learning Technologhy Standards (IEEE LTSC), IMS Global Learning Consortium, Aviation Industry CBT Comité (AICC) y de IMS para algunas de sus funciones, maneja las unidades de contenido con el nombre de SCO (Shareable Content Object) que son simplemente objetos de aprendizaje que cumplen con la especificación SCORM. Lo anterior se describe en el documento SCORM 2004.

Los detalles de la especificación se encuentran en cuatro documentos a los que se da mantenimiento de manera independiente. La versión 1.3 es la más reciente y se conoce como SCORM 2004, los documentos que la componen son:

Libro1*:* *SCORM Overview,* (Thropp S., 2004a). Este documento contiene la historia y los objetivos de ADL y de SCORM, las especificaciones y los estándares necesarios para establecer la definición de SCORM. Además se describe la relación con los otros libros o documentos de la especificación SCORM.

Libro2: SCORM Content Aggregation Model (CAM), (Thropp S., (2004b). Este documento contiene los componentes utilizados en el aprendizaje, el proceso de empacamiento de esos componentes necesarios para el intercambio entre sistemas, la descripción de estos con el fin de permitir la búsqueda, la recuperación y la forma de cómo definir las reglas de secuencia de los componentes. El CAM promueve consistencia en el almacenamiento, etiquetado, empaquetado, intercambio y recuperación de contenidos. Este documento también define las responsabilidades y requisitos para construir contenidos agregados como cursos, lecciones o módulos. Asimismo, contiene información para crear paquetes de contenido, aplicando metadatos y una secuenciación y detalles de navegación. Un paquete de contenido (content package) dentro de un archivo XML con descriptores del objeto y el archivo del objeto. Entre los descriptores se encuentra información para identificar, organizar y procesar el objeto en un LMS.

Libro 3: SCORM Run-Time Environment (RTE), (Thropp S., 2004c). Aquí esta descrita la forma (el puente) para interoperar contenidos de aprendizaje basados en SCO y los LMS. Define los requerimientos de un LMS para administrar actividades de tiempo de ejecución (run-time) en el entorno, como arranque de procesos de contenidos y comunicación entre contenidos, donde se incluyen los elementos del modelo de datos necesarios para transmitir los contenidos al alumno. RTE proporciona el medio para que los contenidos

puedan ser interoperables entre diversas plataformas LMS, sin importar la herramienta con la que fueron creados.

Libro 4: SCORM Sequencing and Navigation (SN), (Thropp S., 2004d). El documento SN describe las políticas que un LMS debe seguir a fin de presentar un aprendizaje específico. El desarrollador del contenido es responsable de definir las políticas a las que el LMS debe seguir. Estas políticas deben estar descritas en la estructura del contenido y se codifican en la sección del paquete del contenido. Después de este proceso se espera que la colección de recursos de aprendizaje puedan ser transferidos con un paquete del entorno de un LMS a otro.

A pesar de los esfuerzos que se han hecho para ampliar sus funciones el SCORM tiene un alcance limitado, porque solo cubre el empaquetamiento y la comunicación del recurso con el LMS, pero ha logrado que su compresión en la forma de uso e implementación sea mucho más sencilla que la del IMS (Instructional Management System), por lo anterior se considera que es el más aplicado para el intercambio de paquetes entre plataformas.

Una vez definido el concepto de OA, nos centraremos en el diseño de los OA en el capítulo 6 para Dominio de Conocimiento de "Programación Estructurada con Algoritmos en Pseudocódigo" representado a través del Grafo Genético (ver Figura 5.1) en la metodología de elaboración de OA. Es importante mencionar que la metodología se describirá en el Anexo I, la cual es la desarrollada por Muñoz Arteaga, et al., (2006) y el modelo ADDIE (Modelo de Análisis, Diseño, Desarrollo, Implementación y Evaluación), Kruse, (2006), es el modelo apropiado para el análisis, diseño, desarrollo, evaluación, implantación y evaluación de materiales y actividades de aprendizaje. A partir del modelo ADDIE, se llega a la metodología ADDIEOA (Análisis, Diseño, Desarrollo, Evaluación e Implantación de un Objeto de Aprendizaje), compuesta por 5 fases, como se describe en el Anexo 1.

3.6 Repositorios de Objetos de Aprendizaje

Como se mencionó anteriormente en este capítulo, una de las limitantes más importantes de SCORM, es que no cuenta con la metodología para la creación de repositorios de Objetos de Aprendizaje en la actualidad, sin embargo el apoyo que nos proporcionan las funciones de uso de los metadatos y la creación de paquetes para mover recursos entre sistemas, nos facilitan las funciones de los ROA, (ADL, 2002).

Actualmente, existen grupos de investigación enfocados al desarrollo de la metodología estandarizada para la construcción de los ROA en las instituciones educativas a nivel mundial con el fin de establecer redes de repositorios de Objetos de Aprendizaje.

Como menciona López, (2005), el interés general de la comunidad está siendo enfocado a conectar y utilizar recursos distribuidos en repositorios heterogéneos, asimismo se está trabajando en corregir las divergencias de los repositorios para canalizarse en el uso de los mismos estándares o al menos el uso, de los que sean compatibles con otros. También se está trabajando en la creación de repositorios federados que se basan en búsquedas propagadas en metadatos distribuidos en distintos servidores. El objetivo final es la interoperabilidad, es decir, que los sistemas tengan la capacidad para trabajar fácilmente con otro u otros. Entre estos sistemas se busca, a través de una normalización tecnológica o procedimental, poder interconectarse para realizar diversas transacciones o actividades, siendo la principal actividad el intercambio de contenidos.

Es importante mencionar que existen diferentes esfuerzos mexicanos para crear repositorios de aprendizaje como el que propone el grupo CUDI (Corporación Universitaria para el Desarrollo de Internet), la Universidad de Guadalajara, COLOR (Colección de Objetos Reusables) de la Universidad Nacional Autónoma de México, y LORAA (Learning Object Repository) de la Universidad Autónoma de Aguascalientes.

Para este proyecto adoptaremos LORAA, por las ventajas que nos proporciona, dado que ya existen materiales diseñados por diferentes universidades como la Universidad Veracruzana y la Universidad de Colima, entre otros, lo cual nos facilitará el intercambio de contenidos didácticos.

Como se ha mencionado, es necesario tomar en cuenta algunas consideraciones de diseño de los repositorios de objetos de aprendizaje (ROA). Para facilitar la formación de los repositorios interoperables con la capacidad de exportar e importar contenidos como paquetes identificados con metadatos, es necesario tomar en cuenta las especificaciones IMS relacionadas con la estandarización de los ROA como son: IMS Digital Repositories Specification, IMS Content Packaging, IMS Learning Resource Metadata e IMS Resource List Interoperability. Asimismo es importante incluir a esta información, la lista de recursos de un OA al IMS Resource List Interoperability para abrir el vínculo de comunicación entre las bibliotecas automatizadas y los ROA.

A pesar de que SCORM no tiene en su modelo el desarrollo de repositorios, éste puede comunicarse con los repositorios gracias al IMS Content Packaging para el intercambio de paquetes, lo que lo hace compatible con las aplicaciones desarrolladas sobre IMS para los Repositorios. Cabe mencionar que ambas

especificaciones operan con un derivado de LOM para la definición de su esquema de metadatos, esto permite el intercambio entre ambas especificaciones. Es importante aplicar el uso de ambas especificaciones, ya que se obtienen mayores ventajas, principalmente porque entre sus especificaciones contempla el almacenamiento de recursos y su conexión con otras aplicaciones, SCORM está siguiendo o adoptando a IMS como referente de su crecimiento, por ello, no es difícil imaginar que si SCORM llegase a desarrollar un documento para la construcción de repositorios, éste sería compatible y, muy posiblemente, basado en la actual IMS Digital Repositories Specification. Así que para efectos de este trabajo y del desarrollo de ROA basados en estándares, las especificaciones IMS resultan ser una opción más sólida, esta propuesta asegura la interoperabilidad con mayor cobertura de un entorno e-learning.

Para mayor información acerca de la Metodología de Diseño de los Objetos de Aprendizaje, consultar el Anexo 1.

Capítulo 4 “Dominio del Sistema: Programación Estructurada con Algoritmos en Pseudocódigo”

4.1 Introducción

El objetivo principal de este capítulo es mostrar los conceptos teóricos básicos de programación estructurada con pseudocódigo que son parte del dominio de conocimientos del SAI. Este material apoya al estudiante para lograr el conocimiento de la programación estructurada, a lo largo de éste se guía al estudiante en los conceptos de diseño de algoritmos con pseudocódigo.

Una de las actividades de los estudiantes de las áreas relacionadas con la ingeniería, es la resolución de problemas usando como herramienta a la computadora. Para poder emplear la computadora para ese fin es necesario que aprendan lenguajes y técnicas de programación. La resolución de un problema exige al menos los siguientes pasos:

1. Definición o análisis del problema.
2. Diseño del algoritmo.
3. Transformación del algoritmo en un programa.
4. Ejecución y validación del programa.

4.2 Conceptos Básicos

4.2.1 Etapas de Diseño

En este tema se abordan por primera vez los conceptos de algoritmo y programación. No debe olvidarse cuando se está estudiando el tema de Programación Estructurada con Algoritmos en Pseudocódigo, antes de iniciar el diseño de un algoritmo (y sin duda cuando se trate de un programa) es necesario haber comprendido y saber resolver el problema por medio de la ayuda de una computadora. Por lo tanto, antes de emprender cualquier diseño es importante contestar las siguientes preguntas acerca del problema que se quiere resolver:

1, ¿Conozco y comprendo cuáles son los datos necesarios para la solución del problema? (Datos de entrada).

2. ¿Conozco y comprendo cuáles son los datos que necesito como solución del problema? (Datos de salida).

3. ¿Cuáles son los métodos con los que cuento para implementar una posible solución al problema?

Una vez que se tiene la respuesta a las preguntas anteriores, se facilitará el diseño del algoritmo y/o programa.

4.2.2 Algoritmo

El objetivo fundamental de este material es enseñar a resolver problemas mediante una computadora. Un programador antes que nada debe saber resolver problemas, por lo que para llegar a ser un programador eficaz se necesita aprender a resolver problemas de un modo riguroso y sistemático, a lo largo de este material hablaremos de la metodología necesaria para resolver problemas mediante programas y a este concepto se define como: **metodología de la programación**. El eje (o corazón) de esta metodología es el concepto de algoritmo. La resolución de un problema exige el diseño de un algoritmo que resuelve el problema.

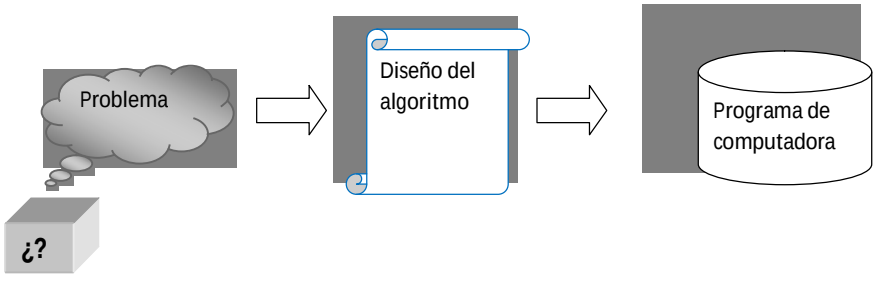


Figura 4.1 Pasos del Diseño de un Programa.

Un *algoritmo* es un método para resolver un problema, debe presentarse como una secuencia ordenada de instrucciones que siempre se ejecutan en un tiempo finito y con una cantidad de esfuerzo también finito. Un algoritmo siempre debe tener un punto de inicio y un punto final, éstos deben ser únicos e identificables.

Los pasos para resolver un problema por medio de la computadora son:

a) ***Diseño del algoritmo*** aquí es necesario describir los pasos de manera ordenada, sin ambigüedades, para resolver el problema. En otras palabras aquí es necesario desarrollar el análisis del problema y el desarrollo del algoritmo.

b) Expresar el algoritmo como un ***programa*** en un lenguaje de alto nivel.

c) Ejecución y validación del programa en la computadora.

Para llegar a la realización de un programa es fundamental el diseño previo de un algoritmo, en otras palabras sin algoritmo no puede existir un programa ver Figura 4.1.

Cabe mencionar que los algoritmos son independientes tanto del lenguaje de programación en que se expresan, como de la computadora donde se ejecutan. Es importante señalar que en cada problema el algoritmo se puede expresar en diferentes lenguajes de programación y ejecutarse en distintas computadoras, sin embargo el algoritmo es el mismo.

En las tecnologías de la información y la programación los algoritmos son más importantes que los lenguajes de programación y las computadoras. Un lenguaje de programación es un medio para expresar un algoritmo y una computadora es sólo un procesador para ejecutarlo. Tanto el lenguaje de programación como la computadora son los medios para obtener un fin: conseguir que el algoritmo se ejecute y se efectúe el proceso correspondiente. Resalta lo anterior la importancia del algoritmo en la computación, el diseño de algoritmos es fundamental. En conclusión, la solución de un problema puede ser expresado mediante un algoritmo. Por la importancia del tema, en este apartado se hace énfasis en el diseño de algoritmos.

Características de un Algoritmo

Todo algoritmo debe cumplir las siguientes características:

Debe ser ***Preciso***; Esto es, debe especificar sin ambigüedad el orden en que se deben ejecutar las instrucciones.

Debe estar ***Definido***; Esto es, cada vez que se ejecute bajo las mismas condiciones, la secuencia de ejecución deberá ser la misma proporcionándonos el mismo resultado.

Debe ser ***Finito***; Esto es, siempre que sea adecuado se realizarán un número finito de instrucciones, en un tiempo finito y requiriendo una cantidad finita de esfuerzo.

Es importante mencionar que en la definición de un algoritmo deben describirse tres partes: Entrada, Proceso y Salida.

Después de definir el concepto de algoritmo es necesario mencionar algunos conceptos básicos que son parte del mismo.

Partes Constitutivas de un Algoritmo

El programador debe establecer el conjunto de especificaciones que debe contener el algoritmo: *entrada, salida y el cuerpo del algoritmo*, que incluirán las técnicas para obtener las salidas a partir de las entradas.

Conceptualmente, un algoritmo deberá ser considerado como una caja negra, como se muestra en la Figura 4.2 La caja negra o el algoritmo de resolución, en realidad, es el conjunto de códice que transforma las entradas del programa (datos) en salidas (resultados).

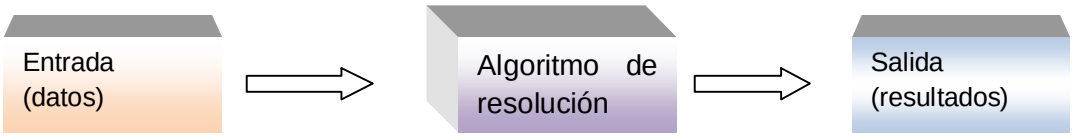


Figura 4.2 Diagrama de Bloques de la Solución de un Problema.

El programador debe establecer de dónde provienen las entradas del algoritmo. Las entradas también son conocidas como entrada de los datos, al proceso de introducir la información de entrada se le llama operación de *lectura* o acción de **leer.** La operación de *salida de datos* se conoce también como *escritura* o acción de **escribir.**

Fases para la Creación de un Algoritmo

Escribir un algoritmo es una actividad que requiere de una buena cantidad de esfuerzo mental y de tiempo. Es necesario comprender el concepto de algoritmo para poder emprender el camino de volver una realidad la construcción de modelos para representar la descripción de fenómenos o de procesos del mundo real.

Esto implica una metodología científica, repetible y comprobable. Se hablará ahora del proceso mental asociado a la construcción de algoritmos. Se observará que las primeras fases de este proceso no difieren demasiado de las equivalentes para casi cualquier otra rama de las ciencias básicas, en el sentido de que constituyen un conjunto de pasos bien específicos que conlleven a la solución.

Las fases en la construcción de un algoritmo, en orden, son:

1. Entender el problema.
2. Hacer análisis del mismo (este paso también lo denominamos *análisis del algoritmo*).
3. Algoritmo (el modelo de solución propuesto).
4. Prueba de escritorio (comprobar que el algoritmo es adecuado a la solución del problema).

El paso uno es fundamental, debemos entender con claridad el problema. La segunda fase, consiste en efectuar un análisis completo del problema, con la finalidad de proponer un modelo para su solución. Debe de ser claro que este modelo no puede existir sin que se especifiquen con claridad todos y cada uno de los componentes estructurales del algoritmo. La estructura de un algoritmo es la forma como están relacionados entre sí sus diversos componentes. Una vez hecho el análisis se procede a convertirlo en el algoritmo, el cual estará escrito en pseudocódigo. El concepto de pseudocódigo será tratado más adelante en esta sección.

Una vez terminada la fase de desarrollo del algoritmo se habrá producido una descripción del modelo propuesto, escrita en seudocódigo. La razón de ser de este paso es disponer de un algoritmo que pueda ser probado mentalmente para ver si es correcto en principio a este proceso, lo podemos llamar la prueba de escritorio.

4.2.3 Otros Componentes

Existen diferentes formas de representar algoritmos, una de las más utilizadas es el lenguaje natural; sin embargo el lenguaje natural no siempre es lo suficientemente preciso para representar el algoritmo. Como una herramienta alternativa tenemos al Pseudocódigo.

Pseudocódigo

El pseudocódigo es *un lenguaje de especificación (descripción) de algoritmos.*

El uso de tal lenguaje hace que el paso de codificación final (esto es, la traducción a un lenguaje de programación) sea relativamente fácil.

El pseudocódigo nació como un lenguaje similar al inglés y era un medio de representar básicamente la estructura de control de programación estructurada. Se considera un *primer borrador*, dado que el pseudocódigo debe de traducirse posteriormente a un lenguaje de programación. El pseudocódigo no puede ser ejecutado por una computadora. La ventaja de usar *pseudocódigo* es que el programador se puede concentrar en la lógica y en las estructuras de control y no preocuparse de la reglas de sintaxis de un lenguaje específico. Es también fácil modificar el pseudocódigo si se descubren errores o anomalías en la lógica del programa, mientras que en muchas ocasiones suele ser difícil el cambio en la lógica una vez que está codificado en un lenguaje de programación. Otra ventaja del pseudocódigo es que puede ser traducido fácilmente a lenguajes estructurados como Pascal, C, FORTRAN 77/90, Ada, C++, etc.

El pseudocódigo original; utiliza para representar las acciones sucesivas palabras reservadas en inglés, similares a sus homónimas en los lenguajes de programación. Por fortuna, aunque el pseudocódigo nació como un sustituto del lenguaje de programación y, por consiguiente, sus palabras reservadas se conservaron o fueron muy similares a las de dichos lenguajes. El uso del pseudocódigo se ha extendido en la comunidad hispana con términos en español como **inicio, fin, parada, leer, escribir, si-entonces-si_no, mientras, fin_mientras, repetir, hasta_que, asignación,** etc. Sin duda, el uso de la terminología del pseudocódigo he facilitado y facilitará considerablemente el aprendizaje y uso diario de la programación. En éste material utilizaremos el pseudocódigo en español.

El pseudocódigo es una forma de representar un algoritmo, mediante sentencias similares al lenguaje natural, pero con una precisión mayor.

Los elementos básicos constitutivos de un algoritmo son:

- *palabras claves o reservadas (***inicio, fin, si –entonces…,** *etc.),*
- *identificadores (*nombre de variables esencialmente),
- *caracteres especiales (*coma, apóstrofe, etc.*),*
- *constantes (*son los valores que no cambian cuando se ejecutan las instrucciones , son valores asignados al principio de los algoritmos*),*
- *variables (*son los valores que sí cambian cuando se ejecutan las instrucciones del algoritmo, debe definirse de qué tipo son éstas),
- *expresiones,*
- *instrucciones.*

Identificadores

Un identificador es un nombre simbólico para una variable, una función, una constante. Un identificador debe cumplir con las siguientes reglas: no deberá ser una palabra reservada, tiene un uso específico y no puede ser utilizada dentro del algoritmo.

Es una secuencia de caracteres de cualquier longitud, con las siguientes reglas a seguir:

1. Debe iniciar con una letra ó "_" y no contener espacios en blanco.
2. Letras, dígitos y caracteres subrayados ("_") están permitidos después del primer carácter del nombre del identificador.
3. Le llamamos caracteres especiales como: el apóstrofe, la coma, el punto y coma, etc.

En síntesis un identificador es un método para nombrar a las celdas de memoria en la computadora, en lugar de tratar de memorizar una dirección.

Son necesarias para darle nombre a: las variables, las constantes, los procedimientos y las funciones.

Palabras Reservadas

Las palabras claves o reservadas son las destinadas para un uso específico en el algoritmo como ejemplo se tienen las palabras reservadas **Algoritmo, imprime**, **lee**, **constante**, etc. Por lo anterior estas palabras reservadas no deben de definirse para otro fin en el algoritmo.

Existen muchas variantes de pseudocódigo, en este material se adoptarán las siguientes normas:

- Todos los algoritmos iniciarán con la sentencia (instrucción) **Algoritmo**. Esto es, una línea con la palabra clave ***Algoritmo*** seguida por el nombre del algoritmo. Por ejemplo:

    Algoritmo EJEMPLO

Enseguida de la sentencia **Algoritmo** se declararán las *constantes* que se manejarán en el programa. La decisión de ***constantes*** se identificará con la palabra clave **constantes**. Por ejemplo:

constante PI 3.1416

constante UNO 1

Después de la declaración de las constantes es necesario representar el desarrollo del algoritmo, aquí es necesario desarrollar los pasos que resuelven el problema, estos pasos estarán dentro del apartado *PRINCIPAL*. Dicho desarrollo se encontrará dentro de las palabras clave Inicio y Fin. Estas palabras clave limitarán un bloque de instrucciones siempre que se requieran. Por ejemplo:

PRINCIPAL

Inicio

.....

Fin

Después de la *palabra* clave ***Inicio*** se declaran las variables necesarias del algoritmo. En este caso el tipo de variables se indicará al principio de la lista; separada por un espacio se colocará la lista de variables, separadas por comas (,). Cada lista incluirá solamente variables de mismo tipo. Ejemplo:

Flotante X, Y, Z

Enteras I, J, K

Para distinguir entre una palabra clave de otras instrucciones del algoritmo se recomienda resaltarlas en **negritas** al imprimir el algoritmo y cuando se use letra manuscrita se sugiere subrayar las palabras clave.

La estructura del pseudocódigo exige normalmente la *indentación* (sangría en el margen izquierdo) de diferentes líneas.

Comentarios

La documentación de un programa es el conjunto de información interna y externa al programa que facilitará su posterior mantenimiento y puesta a punto.

La documentación puede ser interna o externa. La documentación interna es la que se acompaña en el código o programa fuente y se realiza a base de comentarios significativos. Estos comentarios se representan con diferentes notaciones, según el tipo de lenguaje de programación. La documentación externa se acompañará de información ajena al programa y proporcionada al programador.

Representan una ayuda inestimable durante la construcción del algoritmo, siendo imprescindibles para el programador original o las personas que posteriormente analicen el algoritmo. Además de clarificar ideas, los comentarios son también un valioso instrumento de depuración, pues permiten eliminar provisionalmente secciones enteras de código.

Es importante dentro del algoritmo poner comentarios relacionados con el diseño del algoritmo. En cualquier parte del algoritmo podrán colocarse comentarios para hacer aclaraciones o comentarios relacionados con la instrucción u operación que faciliten la interpretación. Los comentarios aparecerán encerrados entre **/"* y *"/.** Ejemplo:

/* Este es un comentario*/

4.2.4 Operadores

Las operaciones son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales. Las mismas ideas son utilizadas en notación matemática tradicional; por ejemplo,

A + (B+3) + $\sqrt{C}$ +B + (B-5) + $\sqrt{C}$

Aquí los paréntesis indican el orden de cálculo y $\sqrt{}$ representa la función raíz cuadrada. Cada expresión toma un valor que se determina tomando los valores de las variables y constantes implicadas y la ejecución de las operaciones indicadas.

Una operación consta de operandos y operadores. Según sea el tipo de objetos que manipulan, las operaciones se clasifican en:

- Asignación.
- Aritméticas.
- Lógicas.
- Relacional.

El resultado de la operación aritmética es de tipo numérico; el resultado de la expresión relacional y de una expresión lógica es de tipo lógico; el resultado de una expresión carácter es de tipo carácter.

Operadores de Asignación

Operadores de asignación Siguiendo la clasificación se presenta *la instrucción de asignación,* esta instrucción es necesaria para representar que una variable va a tomar o asignar un valor dado o definido. Como ejemplo de esto tenemos:

a) s ← 80 la variable s toma el valor de 80.

b) ¿Cuál será el valor que tomará la variable gato tras la ejecución de las siguientes instrucciones?

A ← 12

B ← A

gato ← B

A contiene 12, B contiene 12 y gato contiene 12.

Es importante mencionar que antes de la ejecución de las tres instrucciones, el valor de *A, B* y *gato* es indeterminado. Si se desea darles un valor inicial, habrá que hacerlo explícitamente, incluso cuando este valor sea *0*. Es decir, habrá que definir e inicializar las instrucciones.

A ← 0

B ← 0

gato ← 0

c) ¿Cuál es el valor de la variable *oso* al ejecutarse la instrucción *5*?

k ← 10

B ← 20

oso ← k

A ← B

B ← oso

- en la instrucción 1, k toma el valor de 10
- en la instrucción 2, B toma el valor de 20
- en la instrucción 3, oso toma el valor de k, o sea 10
- en la instrucción 4, A toma el valor anterior de B, o sea 20
- en la instrucción 5, B toma el valor anterior de oso, o sea 10
- después la instrucción 5, oso sigue valiendo 10

d) ¿Cuál es el significado de k ← k + 5 si k tiene el valor actual de 10?

k ← k + 5

Se realiza el cálculo de la expresión N + 5 y su resultado 10 +5 = 15 se asigna a la variable situada a la izquierda, es decir, k tomará un nuevo valor 15.

Se debe de pensar en la variable como en una posición de memoria, cuyo contenido puede variar mediante instrucciones de asignación.

Operadores Aritméticos

Las expresiones aritméticas son análogas a las fórmulas matemáticas (ver Tabla 4.1). Las variables y constantes son numéricas (real o entera) y las operaciones son las aritméticas.

Tabla 4.1 Operadores Aritméticos.

+	suma
-	resta
*	multiplicación
/	división
-,**,^	exponenciación
Div	división entera
Mod	módulo(resto)

Los símbolos +,-,*,**,^(- o **) y las palabras clave **div** y **mod** se conocen como operadores aritméticos (ver Tabla 4.1) . En la expresión: **5+3**, los valores 5 y 3 se denominan *operandos*. El valor de la expresión 5+3 se conoce como *resultado* de la expresión.

Los operandos se utilizan de igual forma que en matemáticas. Por consiguiente, A x B se escribe en un algoritmo como A * B, y ¼ x C como C/4. Al igual que en matemáticas el signo menos juega un doble papel, como resta A-B y como cambio de signo en –A.

No todos los operadores aritméticos existen en todos los lenguajes de programación; por ejemplo, en **FORTRAN** no existe **div** ni **mod**.

El operador exponenciación es diferente según sea el tipo de lenguaje de programación elegido (^, - en **BASIC**, ** en **FORTRAN**).

Los cálculos que implican tipos de datos reales y enteros suelen dar normalmente resultados del mismo tipo, si los operandos lo son también. Por ejemplo, el producto de los operandos reales produce un real (*véase* Tabla 4.2).

Tabla 4.2 Tipos de Resultados.

Operador	Significado	Tipos de Operandos	Tipo de Resultado
-,^,**	Exponenciación	Entero o real	Entero o real
+	Suma	Entero o real	Entero o real
-	Resta	Entero o real	Entero o real
*	Multiplicación	Entero o real	Entero o real
/	División	Real	Real
Div	División Entera	Entero	Entero
Mod	Módulo (resto)	Entero	Entero

Ejemplos:

5 x 7 *se representa por* 5*7

6/4 *se representa por* 6/4

3^7 *se representa por* 3^7

Operadores división "/" y modulo "MOD"

El símbolo / se utiliza para la división. Por Ejemplo

A **/** B

19.0 **/** 6.0, toma el valor **3.1666666**

En forma de operadores resultará la operación anterior:

15 **mod** 6 es 3

Otros ejemplos son:

19.0 **/** 3.0 *equivale a 6.333333*

19 **mod** 6 *equivale a 1*

Por Ejemplo:

Los siguientes ejemplos muestran resultados de expresiones aritméticas.

Expresión	Resultado	Expresión	Resultado
10.5/3.0	3.5	10 **/** 3	3
¼	0.25	18 **/** 2	9
2.0/4.0	0.5	30 **/** 30	1
6/1	6.0	6 **/** 8	0
30/30	1.0	10 **mod** 3	1
6/8	0.75	10 **mod** 2	0

Reglas de Prioridad

Las expresiones que tienen dos o más operandos requieren unas reglas matemáticas que permitan determinar el orden de las operaciones, se denominan reglas de *prioridad o precedencia* y son:

- Las operaciones que están encerradas entre paréntesis se evalúan primero. Si existe diferentes paréntesis anidados (interiores unos a otros), las expresiones más internas se evalúan primero.
- Las operaciones aritméticas dentro de una expresión suelen seguir el siguiente orden de prioridad:
    - Operador exponencial (^, ↑ o bien **).
    - Operadores (*, /, \).
    - Operadores **div** y **mod**.
    - Operadores +, -.

En caso de coincidir varios operadores de igual prioridad en una expresión o subexpresión encerrada entre paréntesis, el orden de prioridad en este caso es de izquierda a derecha. Por Ejemplo:

1) ¿Cuál es el resultado de las siguientes expresiones?

a) 3 + 6 * 14 b) 8 + 7 * 3 + 4 * 6

a) 3 + 6*14 b) 8 + 7 * 3 + 4 * 6

3 + 84 8 + 21 + 24

87 29 + 24

2) *Obtener los resultados de las expresiones:*

-4 * 7 + 2 ^ 3 / 4 – 5

1) -4 * 7 + 8 / 4 -5

2) -28 + 8 / 4 – 5

3) -28 + 2 – 5

4) -26 – 5

-31

3) *Convertir en expresiones aritméticas las siguientes expresiones algebraicas:*

5 **.** (x+y)

$a^2 + b^2$

$$\frac{x+y}{u+\frac{w}{a}}$$

$\frac{x}{y}$ **.** (z + w)

Los resultados serán:

5 * (x+y)

a ^ 2 + b ^ 2

(x + y) / (u + w/a)

x/y * (z + w)

4) Los paréntesis tienen prioridad sobre el resto de las operaciones:

A * (B+ 3)	*la constante 3 se suma primero al valor de B, después ese resultado se multiplica por el valor de A.*
(A * B) + 3	*A y B se multiplican primero y a continuación se suma 3.*
A + (B + C) + D	*esta expresión equivale a A+B+C+D.*
(A + B/C) + D	*equivale a A + B/C + D.*

A * B/C * D *equivale a ((A * B) /C) * D y no a (A * B)/(C * D).*

Operadores Relacionales

Los operadores relacionales o de relación permiten realizar comparaciones de valores de tipo numérico o carácter. Estos operadores permiten comparar los valores de dos expresiones. Los operadores de relación más importantes son los observados en la Tabla 4.3:

Tabla 4.3 Operadores Relacionales.

NOMBRE	SEUDOCÓDIGO	C
Igual a	=	==
Diferente a	!=	!=
Mayor que	>	>
menor que	<	<
Mayor o igual que	>=	>=
menor o igual que	<=	<=

Los operadores de realización se pueden aplicar a cualquiera de los cuatro tipos de datos estándar: *enteros, real, lógico* y *carácter.*

La aplicación a valores numéricos es evidente. Los ejemplos siguientes son significativos:

N1	N2	Expresión Lógica	Resultado
3	6	3<6	Verdadero
0	1	0>1	Falso
4	2	4=2	Falso
8	5	8<=5	Falso
9	9	9>=9	Verdadero
5	5	5<>5	Falso

Para realizar comparaciones de datos de tipo carácter, se requiere una secuencia de ordenación de los caracteres, similar al orden creciente o decreciente. Esta ordenación suele ser alfabética, tanto mayúscula como minúscula, considerándolas de modo independiente. Pero si se consideran caracteres mixtos, se debe recurrir a un código normalizado como el ASCII. Aunque no todas las computadoras siguen el código normalizado en su juego completo de caracteres, sí son prácticamente estándar los códigos de los caracteres alfanuméricos más usuales.

Estos códigos normalizados son:

- Los caracteres especiales #, %, $, (,), +, -, /, …, exigen la consulta del código de ordenación.
- Los valores de los caracteres que representan a los dígitos están en su orden natural. Esto es, ‘0’ <’1’, ‘1’<’2’, …, ‘8’<’9’.
- Las letras mayúsculas A a Z siguen el orden alfabético (‘A’<’B’, ‘C’<’F’, etc.).
- Si existen letras minúsculas, éstas siguen el mismo criterio alfabético (‘a’<’b’, ‘c’<’h’, etc.).

En general, los cuatro grupos anteriores están situados en el código ASCII en orden creciente, Así ‘1’<’A’ y ‘B’<’C’. Sin embargo, para tener completa seguridad será preciso consultar el código de caracteres de su computadora (normalmente, el ASCII (American Standar Code for Informaction Interchange) o bien el EBCDIC (Extended Binary-Coded Decimal Interchange Code), utilizado en computadoras IBM diferentes a los modelos PC y PS/2).

Cuando se utilizan los operadores de relación, con valores lógicos, la constante *false* (falso), es menor que la constante *true* (verdad).

false<true falso<verdadero

true>false verdad>falso

Cuando se utilizan los operadores relacionales = y <> para comparar cantidades numéricas, es importante recordar que la mayoría de los valores reales no pueden ser almacenados exactamente. En consecuencia, las expresiones lógicas formales con comparación de cantidades reales con (=), a veces se evalúan como falsas, incluso aunque estas cantidades sean algebraicas iguales. Así, **(1.0/3.0) * 3.0 = 1.0**, teóricamente es verdadera y, sin embargo, al realizar el cálculo en una computadora se puede obtener .999999….y, en consecuencia, el resultado es falso; esto es debido a la precisión limitada de la aritmética real de las computadoras. Por consiguiente, a veces deberá excluir las comparaciones con datos de tipo real.

Operadores Lógicos

Los operadores lógicos se utilizan cuando necesitamos realizar varias comparaciones en una sola instrucción. Se pueden definir operaciones entre variables lógicas, de las que consideramos las siguientes tres:

1) Negación

Se aplica a una sola expresión lógica, que puede ser una constante o variable lógica o el resultado de una operación lógica. Si el valor de la expresión es Falso, su negación lo hará Verdadero y viceversa. El operador de la negación es el siguiente:

SEUDOCÓDIGO
¬~

La negación se representa mediante la siguiente tabla de verdad:

A	~A
Verdadero	Falso

2) Disyunción

Operación lógica definida para dos operandos. Es verdadera cuando al menos una de las expresiones lo es. En caso contrario es falsa. El operador se escribe como sigue:

SEUDOCÓDIGO
OR

La tabla de verdad correspondiente es la Tabla 4.4:

Tabla 4.4 Tabla de verdad de la disyunción.

A	B	A OR B (se lee "A o B")
Falso	Falso	Falso
Falso	Verdadero	Verdadero
Verdadero	Falso	Verdadero
Verdadero	Verdadero	Verdadero

3) Conjunción

Operación lógica definida para dos operandos. Es verdadera únicamente cuando ambas expresiones lo son. En caso contrario es falsa. El operador se escribe como sigue:

SEUDOCÓDIGO
AND

La tabla de verdad correspondiente es la Tabla 4.5:

Tabla 4.5 Tabla de Verdad de la conjunción.

A	B	A AND B (se lee "A y B")
Falso	Falso	Falso
Falso	Verdadero	Falso
Verdadero	Falso	Falso
Verdadero	Verdadero	Verdadero

4.3 Programa

Un programa de computadora es un conjunto de instrucciones (órdenes dadas a la computadora), que producirán la ejecución de una determinada tarea. En esencia, un programa es un medio para conseguir un fin.

El proceso de programación es por consiguiente un proceso de solución de problemas y el desarrollo de un programa requiere las siguientes fases:

1. Definición y análisis del problema.
2. Diseño de Algoritmos (pseudocódigo).
3. Codificación del programa.
4. Depuración y verificación del programa.

5. Documentación.
6. Mantenimiento.

a) Partes constitutivas de un programa.- Conceptualmente un programa puede ser considerado como una caja negra. La caja negra en realidad es el conjunto de códigos que transforman las entradas del programa (datos) en salidas (resultados), como se muestra en la Figura 4.3.

Figura 4.3 Bloques de un Programa.

El programador debe establecer de dónde provienen las entradas al programa. Las entradas en cualquier caso, provendrán de un dispositivo de entrada (teclado, disco, etc.), al proceso de introducir la información de entrada (datos) en la memoria de la computadora se le conoce como entrada de datos y las salidas de datos se deben presentar en dispositivos periféricos de salida (monitor, impresora, disco, etc.).

b) Instrucciones y tipos de instrucciones.- El proceso de diseño del algoritmo (para su posterior codificación en un lenguaje de alto nivel) consiste en definir las acciones o instrucciones que resolverán el problema. Las acciones o instrucciones se deben escribir en el mismo orden que se han de ejecutar.

Las instrucciones disponibles en cualquier lenguaje de programación dependen del tipo de lenguaje. Por lo anterior, en este apartado estudiaremos las instrucciones –acciones- básicas que se pueden implementar de manera general en un algoritmo representado en pseudocódigo y que esencialmente soportan todos los lenguajes. Dicho de otra forma, las instrucciones básicas son independientes del lenguaje. La clasificación más usual, tomando el punto de vista anterior, es:

1. Instrucciones de inicio/fin.
2. Instrucciones de asignación.
3. Instrucciones de lectura.
4. Instrucciones de escritura.

1. Instrucciones de inicio/fin*.** La instrucción de inicio y fin es muy sencilla esta nos apoya a señalar el ***inicio del algoritmo y el ***fin*** como la conclusión del mismo.

2. Instrucción de lectura. Para continuar con el orden de la clasificación tenemos instrucción de lectura de datos (entrada).

Esta instrucción lee datos de entrada al algoritmo. Y como ejemplo tenemos:

¿Qué desarrollan las instrucciones siguientes?

leer (salario, trabajador, impuesto)

Leer del terminal los valores salario, trabajador, impuesto, guardándolos en la memoria; si los tres números se teclean en respuesta a la instrucción son 1025, 12, 121, significaría que se han asignado a las variables esos valores y equivaldría a la ejecución de las instrucciones.

NUMERO ← 1925

HORAS ← 12

TASA ← 121

3. *Instrucción de Escritura*. Estas instrucciones describen la salida de los valores resultantes de la ejecución de las instrucciones del algoritmo. Como ejemplo tenemos lo siguiente:

Se tiene como resultado de la ejecución de una serie de instrucciones:

A = 100

B = 200

C = 300

escribir (A, B, C)

Aquí se puede notar que la salida de los datos es el siguiente:

100 200 300

que son los valores que corresponde a las variables **A, B** y **C.**

4.3.1 Tipos de Datos

Un dato es la expresión general que describe los objetos con los cuales opera una computadora. Los algoritmos pueden trabajar con varios tipos de (modos) de datos. Los tipos de datos son los siguientes:

- Numéricos (enteros, reales o flotantes).
- Lógicos (booleanos).
- Carácter (carácter, cadenas de carácter).
- Abstractos.

Tipos de Datos Numéricos

Datos numéricos. Este tipo de datos ***numéricos***, es el conjunto de los valores, estos se pueden representar como: tipo numérico entero y tipo numérico real o flotante.

Tipos de Datos Numéricos Enteros

El ***tipo entero*** es un subconjunto finito de los números enteros. Los enteros son números completos, no tiene componentes decimales y pueden ser negativos o positivos, como ejemplo tenemos los valores 45, 17, -18.

Tipos de Datos Numéricos Flotantes(o reales)

El ***tipo real*** consiste de un subconjunto de los números reales. Los números reales siempre tienen un punto decimal y pueden ser positivo o negativos. Un número real consiste de un entero y una parte decimal, como ejemplo tenemos los valores reales:

34.2920 -25.98076

Tipos de Datos Lógicos

Los tipos de ***datos lógicos*** también denominados ***booleanos*** solo pueden tomar uno de los dos valores: *verdadero o falso*. Este tipo de dato se usa para representar las alternativas (*si/no*) en determinadas condiciones.

Tipos de Datos Carácter

El ***tipo carácter*** es el conjunto finito y ordenado de caracteres que la computadora reconoce. Un tipo carácter contiene un solo carácter.

Una ***cadena de caracteres*** es una sucesión de caracteres que se encuentran delimitados por una comilla (apóstrofo) o dobles comillas. La longitud de una cadena de caracteres es el número de ellos comprendidos entre los separadores o limitadores.

Tipos de Datos Abstractos

En este momento es importante dar una vista rápida en el concepto de tipo de datos abstractos como referencia para trabajos futuros.

En el mundo de la programación existen diversos lenguajes que se han ido creando con el paso del tiempo y que se han perfeccionado debido a las necesidades de los programadores de la época a la que pertenecen. Los primeros lenguajes de programación eran de tipo lineal, ya que un programa se recorría desde un punto marcado como Inicio hasta llegar a un punto Fin. Con el tiempo se fueron creando nuevos lenguajes y en nuestros días los más utilizados son los llamados "*Orientados a Objetos*".

Los Lenguajes Orientados a Objetos (LOO) tienen la característica de que no son lenguajes lineales, sino que se forman de diversas funciones, las cuales son llamadas en el orden en que el programa mismo las pide o el usuario determina. Para entender mejor cómo funcionan los Lenguajes Orientados a Objetos, vamos a introducir un concepto fundamental en las Estructuras de Datos denominado Abstracción de Datos y que es parte importante de estos Lenguajes y de la manera en que funciona la mayoría del software comercial de nuestros días.

El concepto de tipo de dato abstracto (TDA "*Abstract Data Types*"), fue propuesto por primera vez hacia 1974 por John Guttag y otros, pero no fue hasta 1975 que por primera vez Liskov lo propuso para el lenguaje CLU.

El lenguaje Turbo Pascal fue determinante para la común aceptación de los TDA´s con la introducción de las unidades. Si bien estas no cumplen con las características básicas de un Tipo de Dato Abstracto, como por ejemplo: la encapsulación de los datos. El lenguaje ADA pudo implementar exitosamente los TDA´s con sus Packages. Vale recordar que estos dos últimos lenguajes soportan formalmente la programación modular.

Con mucha frecuencia se utilizan los términos TDA´s y Abstracción de Datos de manera equivalente, y esto es debido a la similitud e interdependencia de ambos. Sin embargo, es importante definir por separado los dos conceptos.

Como ya se mencionó, los Lenguajes de Programación Orientados a Objetos son lenguajes formados por diferentes métodos o funciones y que son llamados en el orden en que el programa lo requiere, o el usuario lo desea. La abstracción de datos consiste en ocultar las características de un objeto y obviarlas, de manera que solamente utilizamos el nombre del objeto en nuestro programa. Esto es similar a una situación de la vida cotidiana. Cuando yo digo la palabra “perro”, usted no necesita que yo le diga lo que hace el perro. Usted ya sabe la forma que tiene un perro y también sabe que los perros ladran. De manera que yo abstraigo todas las características de todos los perros en un solo término, al cual llamo “perro”. A esto se le llama ‘Abstracción’ y es un concepto muy útil en la programación, ya que un usuario no necesita mencionar todas las características y funciones de un objeto cada vez que éste se utiliza, sino que son declaradas por separado en el programa y simplemente se utiliza el término abstracto (“perro”) para mencionarlo.

En el ejemplo anterior, “perro” es un Tipo de Dato Abstracto y todo el proceso de definirlo, implementarlo y mencionarlo es a lo que llamamos Abstracción de Datos.

Un TDA está caracterizado por un conjunto de operaciones (funciones) al cual le denominaron usualmente como su interfaz pública y representan el comportamiento del TDA; mientras que la implementación como la parte privada del TDA está oculta al programa cliente que lo usa. Todos los lenguajes de alto nivel tienen predefinidos TDA; que son los tipos denominados simples y las estructuras predefinidas, y estos tienen sus interfaces públicas que incluyen las operaciones como la +, -, *, etc. no se necesita conocer cómo actúan tales operadores sobre la representación interna de los tipos definidos, que además, suele ser una implementación bastante dependiente de la máquina sobre la que trabaje el compilador. Lo interesante es que los lenguajes actuales nos van a permitir ampliar los TDA´s predefinidos con otros que serán definidos por el propio programador para adecuar así los tipos de datos a las necesidades de los programas.

Un TDA tendrá una parte que será invisible al usuario la cual hay que proteger y que se puede decir que es irrelevante para el uso del usuario y está constituida tanto por la maquinaria algorítmica que implemente la semántica de las operaciones como por los datos que sirvan de enlace entre los elementos del TDA, es decir, información interna necesaria para la implementación que se esté haciendo para ese comportamiento del TDA. Resumiendo podemos decir, que tanto la implementación de las operaciones como los elementos internos del TDA serán privados al acceso externo y ocultos a cualquier otro nivel.

Algunos ejemplos de utilización de TDA´s en programación son:

- Conjuntos: Implementación de conjuntos con sus operaciones básicas (unión, intersección y diferencia), operaciones de inserción, borrado, búsqueda.
- Árboles Binarios de Búsqueda: Implementación de árboles de elementos, utilizados para la representación interna de datos complejos. Aunque siempre se les toma como un TDA separado son parte de la familia de los grafos.
- Pilas y Colas: Implementación de los algoritmos FIFO (*First Int First Out*) y LIFO *(Last Int First Out)*.
- Grafos: Implementación de grafos; una serie de vértices unidos mediante una serie de arcos o aristas. Estas aristas representan las relaciones entre los conceptos representados por los nodos o vértices.

4.3.2 Variables

Una **variable** es un objeto o partida de datos cuyo valor puede cambiar durante el desarrollo del algoritmo o ejecución del programa. Existen diferentes **tipos de variables** tales como enteras, reales o flotantes, de carácter, lógicas y de cadena. Una **variable** que es de cierto tipo solo puede tomar únicamente valores de ese tipo, como ejemplo tenemos una variable de tipo entero y esta solo puede tomar valores enteros. Si se intenta asignar un valor de un tipo a una variable de otro tipo se producirá un *error de tipo*. Una **variable** se identifica por lo siguientes atributos: nombre que se le asigna y el tipo que describe el uso de la variable.

Los nombres de las **variables**, también conocidos como identificadores, suelen estar formadas por caracteres alfanuméricos, de los cuales el primero normalmente es una letra, se recomienda no utilizar palabras claves o reservadas como nombres de variables, como ejemplo de variables tenemos;

variable entera oso

donde oso es el nombre de la variable y es del tipo entero, esto significa que solo puede recibir valores enteros, por ejemplo oso ← 25.

Las ***expresiones*** son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales como expresiones matemáticas. Cada expresión toma un valor que se determina tomando los valores de las variables y constantes implicadas y la ejecución de las operaciones indicadas.

Una **expresión** consta de *operandos y operadores,* según sea el tipo de objetos que manipulan. Las expresiones se clasifican en: aritméticas, lógicas y carácter.

El resultado de la expresión aritmética es de tipo numérico; el resultado de la expresión relacional y de una expresión lógica es de tipo lógico; el resultado de una expresión carácter es de tipo carácter.

En síntesis los tipos de datos primitivos (ver Figura 4.4) se clasifican en:

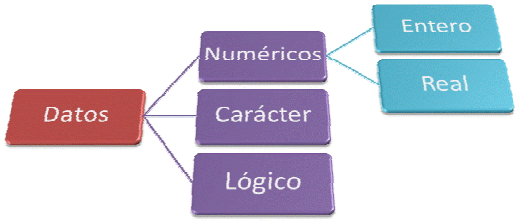


Figura 4.4 Tipos de Datos Primitivos.

4.3.3 Estructuras de Control

En esta sección se mostrarán algunas de las herramientas existentes para escribir algoritmos, una vez que el problema se ha comprendido y que se ha efectuado el análisis, no importando que sea elemental.

Las estructuras de control son las formas que existen para dirigir el flujo de acciones que debe de seguir el algoritmo para resolver el problema. Además constituyen la herramienta que nos apoya a la construcción y representación de las secuencia de actividades a desarrollar en el algoritmo.

Existen tres *estructuras de control básicas ó primitivas* y combinándolas se puede escribir cualquier algoritmo. Estas estructuras primitivas son: *la Secuencia, la Bifurcación Condicional y el Ciclo.*

Las estructuras de control básicas, que describiremos a continuación son: 1) *Secuencia*, 2*) Iteración: Condicional (mientras y repite) y No-Condicional (progresión aritmética), y 3) Selección (sencilla: SI_ENTONCES_SINO y múltiple: CASO)*, mientras que las estructura de datos más comunes son los *arreglos (o vectores), listas, cadenas, pilas y árboles*, también se emplean – aunque en este material no se estudian. Como componentes no estructurales de un programa se puede mencionar, en orden creciente de complejidad, los *enunciados, instrucciones, funciones, procedimientos (subrutinas) y módulos*.

Se considera elementos no estructurales a los antes mencionados, a que su aparición dentro del programa obedece a razones guiadas por los componentes que sí lo son: las estructuras de datos y de control. O lo que es igual, lo primero que hay que definir al construir un programa son precisamente sus elementos estructurales.

Las estructuras de control determinan la secuencia en que deben ejecutarse las instrucciones de un algoritmo.

Secuencia

La estructura de control más simple es la Secuencia. Esta estructura permite que las instrucciones que la constituyen se ejecuten una tras otra en el orden en que se listan. Por ejemplo, considérese el siguiente fragmento de un algoritmo representado por la Figura 4.5.

Figura 4.5 Estructura de Control de Secuencia.

En este fragmento se indica que se ejecute la operación 1 y a continuación la operación 2.

Pero no todos los procesos pueden ser descritos con esta estructura de control. Se hace necesaria una que permita tomar decisiones sencillas. Esta nueva construcción primitiva se llama **Selección**.

Selección

La estructura control de Selección permite evaluar, durante la ejecución, una condición booleana, para decidir cuál de dos caminos elegir. Una condición se llama booleana cuando puede adquirir únicamente dos valores de verdad: verdadero o falso.

Selección Simple

Esta estructura de control permite elegir entre dos opciones o alternativas posibles, si la condición se cumple o no.

Véase la Figura 4.6 del siguiente fragmento de algoritmo:

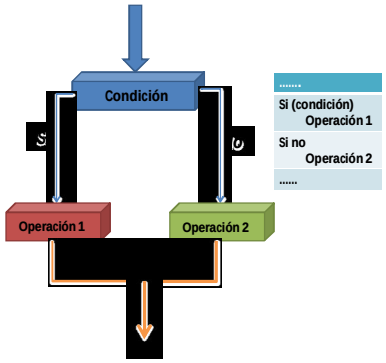


Figura 4.6 Estructura de Control Selección Simple.

Selección Múltiple

Con frecuencia en la práctica, es necesario que existan más de dos elecciones posibles (por ejemplo, en la resolución de la ecuación de segundo grado existen tres posibles alternativas o caminos a seguir, dependiendo si el discriminante es nulo, positivo o negativo). Este problema se podría resolver

por estructuras alternativas simples o dobles, anidadas o en cascada; sin embargo este método si el número de alternativas es grande puede plantear serios problemas de escritura del algoritmo y de legibilidad.

La estructura de decisión múltiple evaluará una expresión que podrá tomar **n** valores distintos, **1, 2, 3, 4,…, n**. Según se elija a alguno de estos valores de la condición se realizará una de las **n** acciones, según se muestra en la Figura **4.7**.

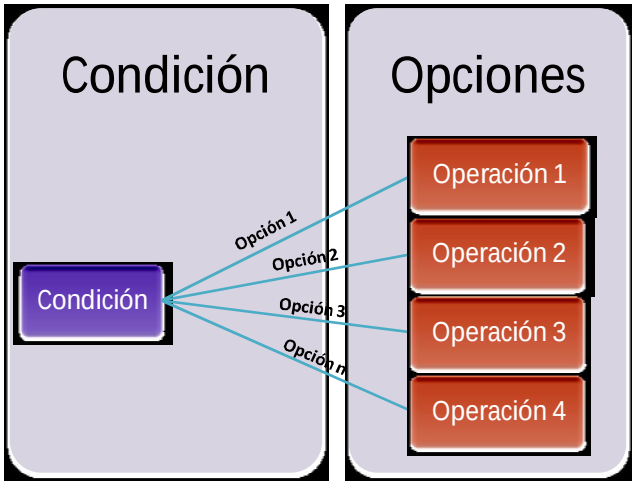


Figura 4.7 Estructura de Control Selección Múltiple.

Iteración

Una tercera primitiva que completará el conjunto de estructuras de control y con las cuales será posible construir cualquier algoritmo es la estructura de control de *Iteración Condicional*.

La *Iteración* o también llamados *ciclos* son estructuras de control que evalúan cierta condición booleana. Si el resultado es verdadero, ejecuta una preposición y continúa de esta manera mientras la condición siga siendo verdadera. Está claro que el ciclo se romperá cuando la condición deje de ser verdadera y se vuelva falsa. Es posible, incluso, que el ciclo nunca se efectué, si desde el principio la condición es falsa, existen diferentes tipos de ciclos como se muestran a continuación.

Iteración Condicional (Mientras)

Las estructuras de iteración condicionales son aquellas que repiten una secuencia de instrucciones un número determinado de veces controladas por el cumplimiento de una condición.

Este ciclo repite una operación, mientras la condición booleana sea verdadera. Por ejemplo ver la Figura 4.8.

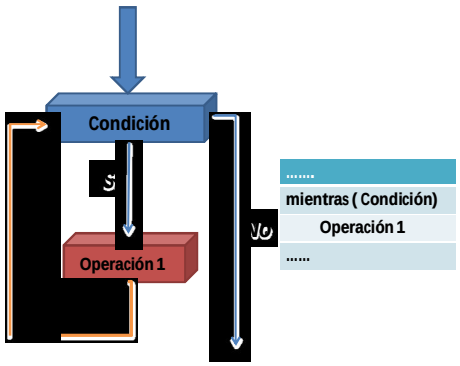


Figura 4.8 Estructura de Control de Iteración Condicional (Mientras).

Iteración Condicional Repite_Hasta (Hacer-Mientras)

En esta estructura de control ejecuta al menos una vez la operación y después se evalúa la condición booleana, como se aprecia en la Figura 4.9. Si es verdadera la condición, la operación se ejecuta de nuevo, y así sucesivamente hasta que la condición booleana sea falsa.

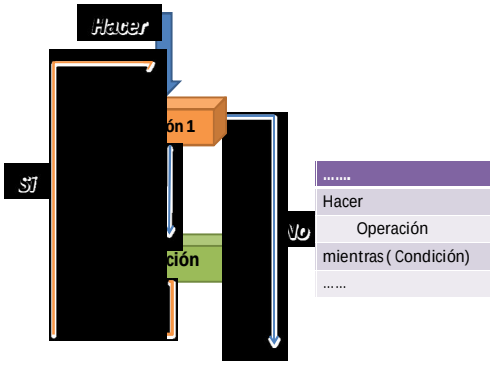


Figura 4.9 Estructura de Control de Iteración Condicional (Hacer-Mientras).

Iteración No Condicional de Progresión Aritmética (Desde)

En este ciclo se ejecuta una operación un cierto número de veces, especificando en un contador el incremento unitario, desde un *valor inicial* hasta un *valor final* que marcará la condición de salida del ciclo. Como se aprecia en la Figura 4.10.

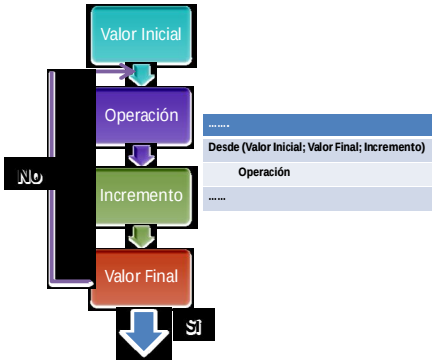


Figura 4.10 Estructura de Control de Iteración No Condicional de Progresión Aritmética (Desde).

Una observación primordial es que cada una de estas estructuras de control tiene un *solo punto de entrada* y *uno de salida*. Esto servirá para construir algoritmos completos uniendo salidas con entradas, para formar cadenas de preposiciones estructuradas en secuencia.

Es posible combinar las estructuras de control de secuencia, selección e iteración condicional, utilizando para tal fin la secuencia, la selección y la iteración condicional. Es decir es posible combinar las estructuras entre si, en caso de ser necesario para construir el algoritmo.

En el caso que una estructura de control tenga más de una instrucción es necesario delimitar en forma adecuada el alcance de las proposiciones o sentencia de la estructura, esto se logra a través de las palabras reservadas ***inicio*** y ***fin***, así mismo se recomienda **identar** las instrucciones para hacer una clara distinción del alcance de las instrucciones.

La estructura del pseudocódigo exige normalmente la *identación* (sangría en el margen izquierdo) de diferentes líneas.

Como ya se dispone de las estructuras básicas de control. Es momento de mostrar algunos ejemplos de uso de las estructuras de control.

Ejemplos de Aplicación de Estructuras de Control

EJEMPLO 1

El siguiente ejemplo de algoritmo calcula la conversión de grados a radianes. El dato de entrada es el grado a calcular en radianes, es importe mencionar que este algoritmo solo calcula un dato de entrada y solo obtiene un dato de salida. Este algoritmo ejemplifica el uso de la estructura de control de secuencia.

Algoritmo GRADOS A RADIANES

/* Este algoritmo convierte de grados a radianes.

Ejemplo de una Estructura Secuencial. */

constante PI 3.1416

/*en esta parte del algoritmo se definen las constantes*/

PRINCIPAL

Inicio

flotante GRAD, RAD ;

/* en esta parte del algoritmo se definen las variables del algoritmo*/

leer (GRAD); **/* entrada o lectura de los datos ·*/**

RAD ← GRAD * PI /180;

/* El símbolo * indica multiplicación y el símbolo ← se define como:

asignación de un valor una o una operación a una variable*/

imprimir (RAD) ; **/* salida o impresión de los datos */**

Fin

EJEMPLO 2

En este ejemplo se calcula la ecuación de segundo grado. El algoritmo en su diseño se está aplicando una estructura de control de selección y secuencia.

Como muestra el ejemplo, este algoritmo calcula la ecuación de segundo grado, tomando en cuenta los casos cuando el producto [(b * b) – 4 * a * c] es un valor positivo, si este valor es negativo se saca el valor absoluto del producto para evitar en este caso los valor a calcular la raíz cuadrada sea una raíz imaginaria.

Algoritmo SEGUNDO GRADO

/* Calcula las raíces reales de la ecuación de segundo grado:

$aX^2 + bX + c = 0$.

Ejemplo de Estructura de Bifurcación Condicional. */

PRINCIPAL

Inicio

flotante a, b, c, d, X1, X2, RE, IM ;

imprimir (“DAME LOS VALORES DE a, b, c”) ;

leer (a, b, c) ;

d← (b * b) – 4 * a * c ;

si (d > 0)

Inicio

/* La función sqrt calcula la raíz cuadrada */

X1 ← (-b + **sqrt** (d)) / (2 * a);

X2 ← (-b - **sqrt** (d)) / (2 * a);

imprimir (“LAS RAICES SON:”) ;

imprimir (X1 , X2) ;

Fin

sino

Inicio

/* La función fabs calcula el valor absoluto y la función sqrt calcula la raíz cuadrada */

d← **fabs** (d) ;

RE ← -b / (2 * a);

IM ← **sqrt** (d) / (2 * a);

imprimir (“ LAS RAICES SON IMAGINARIAS Y SON:”) ;

imprimir (RE, IM) ;

Fin

Fin

EJEMPLO 3

En este ejemplo se desea calcular la potencia de un número cualquiera, los datos a calcular: la base y la potencia están dados por teclado, aquí se aplican las estructuras de control de iteración (Ciclo mientras) y la secuencia.

Algoritmo POTENCIA

/* Calcula el número X elevado a la n.

Ejemplo de Estructura Iterativa (ciclo Mientras.) */

PRINCIPAL

Inicio

flotante X, pot, n, k

imprimir (“Dame el valor de la base y del exponente”) ;

leer (X, n) ;

k ← 0 ;

pot ← 1 ;

mientras (k < n)

Inicio

pot ← pot * X ;

k ← k + 1 ;

Fin

imprimir (“El resultado es” pot) ;

Fin

EJEMPLO 4

Este ejercicio calcula de forma iterativa el área de un circulo, hasta que encuentre una área mayor al área dada como máxima, también se muestra(o se imprimen) los valores de radio (rad) y el incremento al radio (inc), estos datos de entrada son solicitados desde teclado. En este algoritmo se aplican las estructuras de control de iterativas y secuenciales.

Algoritmo AREA MAXIMA

/* Calcula cuando el área de un círculo es mayor a cierto valor dado. Despliega el valor del radio para el cual todavía es menor y el área ocupada. Ejemplo de estructura Iterativa (Ciclo Hacer_Mientras.) */

constante pi 3.1416

PRINCIPAL

Inicio

flotante rad, inc, amax, área

imprimir (“Radio Inicial”)

leer (rad)

imprimir (“Incremento del Radio”)

leer (inc)

imprimir (“Área Máxima”)

leer (amax)

hacer

Inicio

area←pi*rad*rad

rad←rad+inc

Fin

mientras (amax > área)

rad ←rad-inc

area ← pi*rad*rad

imprimir (“Para el radio”, rad)

imprimir (“se tiene un área de “, área)

imprimir (“que es menor el área máxima de “, amax)

Fin

EJEMPLO 5

Este ejercicio calcula el promedio de un alumno con tres calificaciones para después calcular el promedio del grupo. En este ejercicio se aprecia el uso de las estructuras de control Iterativas (ciclo desde) y secuenciales. Los datos de entrada son: N, CALIF, están dadas desde teclado. Cabe mencionar que en este algoritmo se utilizo una estructura iterativa dentro de otra estructura iterativa (un ciclo dentro de otro ciclo) para calcular primero el promedio de las tres calificaciones para cada alumno e imprimirlo, con el fin de calcular el promedio de todos los alumnos.

Algoritmo PROMEDIOS

/* Para un grupo con N alumnos se calcula el promedio de sus calificaciones, teniendo cada alumno tres calificaciones diferentes.

Ejemplo de Estructura Ciclo Desde*/

PRINCIPAL

Inicio

flotante N, i, j, SUM CALIF, PROM

imprimir (“DAME EL NUMERO DE ALUMNOS”)

leer (N)

desde (i←1; i < = N; i ← i +1)

Inicio

SUM←0

desde (j ←1; j < = 3; j← j + 1)

Inicio

imprimir (“DAME LA CALIFICACION”, j)

leer (CALIF)

SUM←SUM + CALIF **/* aquí se guardan los valores de las calificaciones SUM */**

Fin

PROM←SUM / 3;

imprimir (“EL PROMEDIO DEL ALUMNO ES”, PROM)

Fin

Fin

4.3.4 Abstracción (procedimental y funcional)

La abstracción, una de las herramientas que nos apoya en la solución de un problema, es un mecanismo fundamental para la comprensión de problemas y fenómenos que poseen una gran cantidad de detalles, su idea principal consiste en manejar un problema, fenómeno, objeto, tema o idea como un concepto general, sin considerar la gran cantidad de detalles que estos puedan tener. El proceso de abstracción presenta dos aspectos complementarios.

1. Destacar los aspectos relevantes del objeto.
2. Ignorar los aspectos irrelevantes del mismo (la irrelevancia depende del nivel de abstracción, ya que si se pasa a niveles más concretos, es posible que ciertos aspectos pasen a ser relevantes).

De modo general podemos decir que la abstracción permite establecer un nivel jerárquico en el estudio de los fenómenos, el cual se establece por niveles sucesivos de detalles. Generalmente, se sigue un sentido descendente de detalles, desde los niveles más generales a los niveles más concretos.

Por ejemplo: los Lenguajes de Programación de alto nivel permiten al programador abstraerse del sin fin de detalles de los lenguajes ensambladores. Otro ejemplo, la memoria de la computadora es una estructura unidimensional formada por celdas y sin embargo se trabaja como si fuera única. La abstracción brinda la posibilidad de ir definiendo una serie de

refinamientos sucesivos a la tarea de diseño de algoritmos y cuando dice refinamientos sucesivos se refiere a la estrategia que se utiliza para descomponer un problema en subproblemas. Conforme evoluciona el diseño de software a cada nivel de módulos se representa un refinamiento en el nivel de abstracción. Esto es, incluir detalles que fueron obviados en un nivel superior, en un nivel más bajo de la jerarquía.

Veamos los diferentes tipos de abstracción que podemos encontrar en un programa:

Abstracción funcional: crear procedimientos y funciones e invocarlos mediante un nombre donde se destaca qué hace la función y se ignora cómo lo hace. El usuario sólo necesita conocer la especificación de la abstracción (el qué) y puede ignorar el resto de los detalles (el cómo).

Abstracción de datos:

- **Tipo de datos**: proporcionado por los leguajes de alto nivel. La representación usada es invisible al programador, al cual solo se le permite ver las operaciones predefinidas para cada tipo.
- **Tipos definidos**: por el programador que posibilitan la definición de valores de datos más cercanos al problema que se pretende resolver.
- **TDA**: para la definición y representación de tipos de datos (valores + operaciones), junto con sus propiedades.
- **Objetos**: Son tareas de diseño de algoritmos a los que se añade propiedades de reutilización y de compartición de código.

Si profundizamos más al mundo de la programación y sus conceptos, existen dos de estos conceptos que no se deben confundir, ellos son: tipo de datos y estructura de datos.

Un tipo de dato, en un lenguaje de programación, define un conjunto de valores que una determinada variable puede tomar, así como las operaciones básicas sobre dicho conjunto. Ahora veamos cómo se van relacionando estos conceptos. Los tipos de datos constituyen un primer nivel de abstracción, ya que no se tiene en cuenta cómo se implementan o se representan realmente la información sobre la memoria de la máquina. Para el usuario, el proceso de implementación o representación es invisible.

Veamos entonces que son las estructuras de datos. Las estructuras de datos son colecciones de variables, no necesariamente del mismo tipo, relacionadas entre sí de alguna forma. Las estructuras de datos están caracterizadas por el

tipo de dato de los elementos guardados en la estructura y por la relación definida sobre estos elementos.

Al nivel de las estructuras de datos son totalmente irrelevantes las operaciones sobre un elemento en particular, solamente tienen carácter relevante las operaciones que envuelvan la estructura de forma global.

Abstracciones Procedimentales (subrutinas)

En ocasiones dentro de la programación se presentan problemas difíciles de resolver, provocando que un algoritmo (o programa) se complique demasiado en su elaboración. Y aún más en el propio proceso de cálculo.

Cuando se tiene que resolver un problema complejo se recomienda descomponer el problema en subproblemas. Esta estrategia facilitará la construcción del algoritmo de solución del problema.

A esta técnica de elaboración y programación se le conoce como Diseño Descendente.

Que consiste de descomponer en "subprogramas", elaborados de la misma manera que el principal, desarrollando éstos unas tareas más sencillas y fáciles de comprender y programar.

Un *procedimiento o subrutina* es un programa que ejecuta un proceso específico. Ningún valor está asociado con el nombre del procedimiento; por consiguiente, no puede ocurrir en una expresión. Un procedimiento se llama escribiendo su nombre, por ejemplo, CUENTA, para indicar que un procedimiento llamado CUENTA se va a usar. Cuando se manda llamar al procedimiento, los pasos que lo definen se ejecutan y cuando termina devuelve el control al programa que lo llamó.

La *declaración de un procedimiento* es similar a la de funciones.

procedimiento *nombre [(lista de parámetros formales)]*

Los parámetros formales tienen el mismo significado que en las funciones; los parámetros variables —en aquellos lenguajes que los soportan, por ejemplo, Pascal— están precedidos cada uno de ellos por la palabra **var** para designar que ellos obtendrán resultados del procedimiento en lugar de los valores actuales asociados a ellos.

El procedimiento se llama mediante la instrucción:

[**llamar_a**] *nombre* *[(lista de parámetros*

La palabra llamar_a es opcional y su existencia depende del lenguaje de programación. El ejemplo siguiente ilustra la definición y uso de un procedimiento para realizar la división de dos números y obtener el cociente y el resto.

Variables enteras: DIVIDENDO

DIVISOR

COCIENTE

Procedimiento

procedimiento división (E entero: Dividendo, Divisor; S entero: Cociente, Resto)

Inicio

Cociente ← Dividendo **div** Divisor

Resto ← Dividendo – Cociente * Divisor

fin_procedimiento

Algoritmo principal

algoritmo Aritmética

var

entero: M, N, P, Q, S, T

Inicio

leer(M, N)

llamar_a division (M, N, P, Q)

escribir(P, Q)

llamar_a division (M * N -4, N + 1, S, T)

escribir(S, T)

Fin

Sustitución de argumentos/parámetros

La lista de parámetros, *formales* en el procedimiento o *actuales* (reales) en la llamada, se conoce como *lista de parámetros*, pueden ser de Entrada (**E**), Salida (**S**) o Entrada/Salida (**E/S**).

procedimiento *demo*

fin_procedimiento

o bien:

procedimiento *demo (lista de parámetros formales)*

y la instrucción que llama

llamar_a ***demo (lista de parámetros actuales)***

Cuando se llama al procedimiento, cada parámetro formal toma como valor inicial el valor del correspondiente parámetro actual. En el ejemplo siguiente se indica la sustitución de parámetros y el orden correcto.

algoritmo Demo

//definición del procedimiento

entero: años

real: números, tasa

inicio

...

llamar_a calculo (numero, años, tasa)

...

fin

procedimiento calculo (S real: p1; E entero: p2; E real: p3)

inicio

p3 … p1 … p2 … p2

fin_procedimiento

Las acciones sucesivas a realizar son las siguientes:

1. Los parámetros reales sustituyen a los parámetros formales.
2. El cuerpo de la declaración del procedimiento se sustituye por la llamada del procedimiento.
3. Por último, se ejecutan las acciones escritas por el código resultante.

Las variables utilizadas en los programas principales y subprogramas se clasifican en dos tipos:

• Variables Locales.

• Variables Globales.

Variables Locales

Una *variable local* es aquella que está declarada y definida dentro de un subprograma, y su valor solo es conocido en ese subprograma, además es distinta de las variables con el mismo nombre declaradas en cualquier parte del programa principal. *El significado de una variable se confina al procedimiento en el que está declarada*. Cuando otro subprograma utiliza el mismo nombre se refiere a una posición diferente en memoria. Se dice que tales variables son locales en el subprograma en el que están declaradas.

Variables Globales

Una *variable global* es aquella que está declarada en el programa o algoritmo principal, del que dependen todos los subprogramas.

Ventajas entre variables globales y locales

La parte del programa/algoritmo en que la variable se define se conoce como *ámbito* (*scope*, en inglés).

El uso de las variables locales tiene muchas ventajas. En particular hace a los subprogramas independientes, con la comunicación entre el programa principal y los subprogramas manipulados estructuralmente a través de lista de parámetros. Para utilizar un procedimiento solo necesitamos conocer lo que hace y no tenemos que estar preocupados por su diseño, es decir, en algunos casos, cómo están programados.

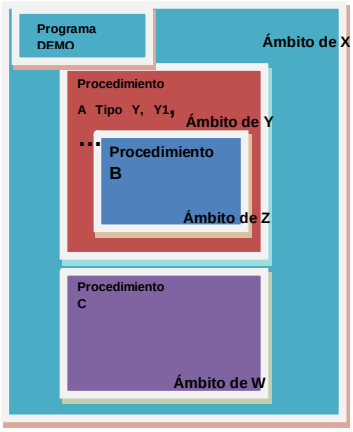


Figura 4.11 Ámbito de Identificadores.

Esta característica hace posible dividir grandes proyectos en piezas más pequeñas independientes (ver Figura 4.11). Cuando diferentes programadores están implicados, ellos pueden trabajar de manera independientemente.

A pesar del hecho importante de los subprogramas independientes y las variables locales, la mayoría de los lenguajes proporcionan algún método para tratar ambos tipos de variables.

Una variable local a un subprograma no tiene ningún significado en otros subprogramas. Si un subprograma asigna un valor a una de sus variables locales, este valor no es accesible a otros programas, es decir, no pueden utilizar este valor. A veces es necesario que una variable tenga el mismo nombre en diferentes subprogramas.

Por el contrario, las variables globales tienen la ventaja de compartir información de diferentes subprogramas sin una correspondiente entrada en la lista de parámetros.

En un programa sencillo con un subprograma, cada variable u otro identificador es o bien local al procedimiento o global al programa completo. Sin embargo, si el programa incluye procedimientos que engloban a otros procedimientos (*procedimientos anidados)*, entonces la noción de global/local es algo más complicado de entender.

El *ámbito* de un identificador (variables, constantes, procedimientos) es la parte del programa donde se conoce el identificador. Si un procedimiento está definido localmente a otro procedimiento, tendrá significado solo dentro del ámbito de ese procedimiento. A las variables les sucede lo mismo; si están definidas localmente dentro de un procedimiento, su significado o uso se confina a cualquier función o procedimiento que pertenezca a esa definición.

La Figura 4.11 muestra un esquema de un programa con diferentes procedimientos, algunas variables son locales y otras globales. En la citada figura se muestra el ámbito de cada definición.

Los lenguajes que admiten variables locales y globales suelen tener la posibilidad explícita de definir dichas variables como tales en el cuerpo del programa o, lo que es lo mismo, definir su ámbito de actuación, para ello se utilizan las cabeceras de programas y subprogramas, con lo que se definen los ámbitos.

Las variables definidas en un ámbito son accesibles en el mismo, es decir, en todos los procedimientos anteriores (ver Figura 4.12).

Variables Definidas en	**Accesibles desde**
A	A,B,C,D,E,F,G
B	B,C
C	C
D	D,E,F,G
E	E,F,G
F	F
G	G

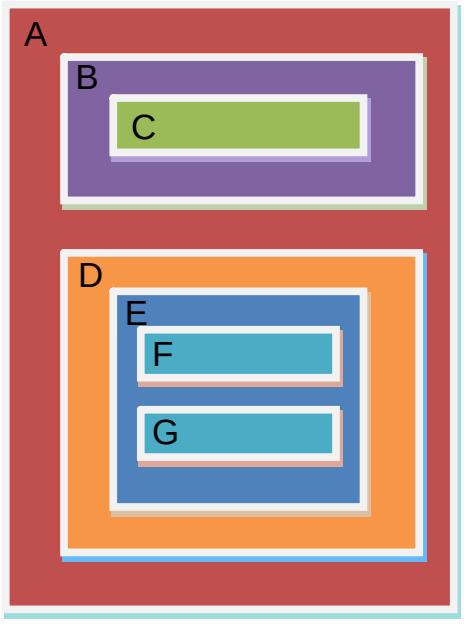


Figura 4.12 Ámbito de Definición de Variables.

Ejemplo

La función (signo) realiza la siguiente tarea; dado un número real x, si x es 0, entonces se devuelve un 0; si x es positivo, se devuelve 1, y si x es negativo, se devuelve el valor -1.

La declaración de la **funcion** es:

```
    entero funcion signo (E real: x)
var entero: s
inicio
  //valores de signo: +1, 0, -1
  si x = 0 entonces s ← 0
  si x > 0 entonces s ← 1
  si x < 0 entonces s ← -1
  devolver (s)
fin funcion
```

Antes de llamar a la función, la variable (S), como se declara dentro del subprograma, es local al subprograma y sólo se conoce dentro del mismo. Veamos ahora un pequeño algoritmo donde se invoque la función.

```
algoritmo SIGNOS
var
  entero: a,b,c
  real: x,y,z
inicio
  x ← 5.4
  a ← signo (x)
  y ← 0
  b ← signo(y)
  z ← 7.8975
```

c ← signo (z-9)

escribir (‘Las respuestas son’, a, ‘ ‘ , b, ‘ ’, c)

fin

Si se ejecuta este algoritmo, se obtienen los siguientes valores:

x = 5.4 **x es el parámetro actual de la primera llamada a signo(x)**

a = signo (5.4) **a toma el valor de 1**

y = 0

b = signo (0) **b toma el valor de 0**

z = 7.8975

c = signo (7.8975-9) **c toma el valor de -1**

La línea escrita al final será:

Las respuestas son: 1 0 -1

Ejemplo:

algoritmo DEMOX

var entero: A, X, Y

inicio

x ← 5

A ← 10

y ← F(x)

escribir (x, A, y)

fin

entero **funcion** F(E entero: N)

var

entero: X

inicio

A ← 5

X ← 12

devolver (N+A)

fin_funcion

A la variable global A se puede acceder desde el algoritmo y desde la función. Sin embargo, X identifica a dos variables distintas: una local al algoritmo y sólo se puede acceder desde él y otra local a la función.

Al ejecutar al algoritmo se obtendrían los siguientes resultados:

X = 5

A = 10

Y = F(5) **invocan a la función F(N) se realiza un paso de parámetro formal N**

A = 5 **se modifica el valor de A en el algoritmo principal por ser A global**

X = 12 **no se modifica el valor de X en el algoritmo principal porque X es local**

F = 5 + 5 = 10 **se pasa el valor del argumento X(5) a través del parámetro N**

Y=10

Se escribirá la línea:

5 5 10

ya que X es el valor de la variable local X en el algoritmo; A, el valor de A en la función, ya que se pasa este valor al algoritmo; Y es el valor de la función F(X).

Abstracciones Funcionales

Las funciones son herramientas de programación que apoyan a la resolución de problemas, pero su alcance está muy limitado. Es muy común que se

requieran que los subprogramas calculen *varios* resultados *en vez de uno solo*. Para estos casos una *función* no es apropiada y se necesita disponer del otro tipo de subprograma: *el procedimiento o subrutina.*

Los procedimientos y las funciones son subprogramas con diseño y misión similares. Las diferencias entre estos subprogramas son:

- Un procedimiento es llamado desde el algoritmo principal mediante su nombre y una lista de parámetros actuales o bien con la instrucción llamar_a. Al llamar al procedimiento se detiene momentáneamente el programa y el control pasa al procedimiento llamado. Después de que las acciones del procedimiento se ejecutan, se regresa a la acción inmediatamente siguiente a la que se llamó.
- Las funciones devuelven un valor, los procedimientos pueden devolver 1,0 o *n* valores y en forma de lista de parámetros.
- El procedimiento se declara igual que la función, pero su nombre no está asociado a ninguno de los resultados que obtiene.

Comunicación con Subprogramas

Cuando un programa llama a un subprograma, la información se comunica a través de la lista de parámetros y se establece una correspondencia automática entre los parámetros formales y actuales.

Los parámetros actuales son <<sustituidos>> o <<utilizados>> en lugar de los parámetros formales.

La declaración del subprograma se hace con

```
procedimiento nombre (clase tipo_de_dato: F1
                      clase tipo_de_dato: F2
                      ……
                      clase tipo_de_dato :Fn)
 .
```

y la llamada al subprograma con:

llamara_a nombre (A1, A2, … , An)

donde F1, F2, …, Fn son los parámetros formales y A1, A2, …, An, son los parámetros actuales o reales.

Las clases de parámetros podrían ser:

- (E) Entrada
- (S) Salida
- (E/S) Entrada/Salida

Existen dos métodos para establecer la correspondencia de parámetros:

1. *Correspondencia posicional.* La correspondencia se establece aparejando los parámetros reales y formales según su posición en las listas: así, Fi se corresponde con Ai, donde i= 1, 2, …, n. Este método tiene algunas desventajas de legibilidad cuando el número de parámetros es grande.
2. *Correspondencia por el nombre explícito*, también llamado *método de paso de parámetros por nombre.* En este método, en las llamadas se indica explícitamente la correspondencia entre los parámetros reales y formales. Este método se utiliza en Ada. Un ejemplo sería:

```
SUB(Y => B, X => 30);
```

que hace corresponder al parámetro actual B con el formal Y, y el parámetro actual 30 con el formal X durante la llamada de SUB.

Por lo general, la mayoría de los lenguajes usan exclusivamente la correspondencia posicional y ése será el método empleado en este libro.

Las cantidades de información que pueden pasarse como parámetros son *datos, tipos* en los lenguajes que admiten su declaración y *subprogramas.*

Paso de Parámetros

Existen diferentes métodos por la *transmisión o el paso de parámetros a subprogramas.* Es preciso conocer el método adoptado por cada lenguaje, ya que la elección puede afectar a la semántica del lenguaje. Dicho de otro modo, un mismo programa puede producir diferentes resultados bajo diferentes sistemas de paso de parámetros.

Los parámetros pueden ser clasificados como:

Entradas.- las entradas proporcionan valores desde el programa que llama y que (**E**), se utilizan dentro de un procedimiento. En los subprogramas *función*, las entradas son los argumentos en el sentido tradicional;

Salidas.- las salidas producen los resultados del subprograma; de nuevo si se (**S**) utiliza el caso de una función, éste devuelve un valor calculado por dicha función, mientras que con procedimientos pueden calcularse cero, una o varias salidas;

Entradas/Salidas.- un solo parámetro se utiliza para mandar argumentos a un programa (**E/S**) y para devolver resultados.

Desgraciadamente, el conocimiento del tipo de parámetros no es suficiente para caracterizar el funcionamiento; por ello, examinaremos los diferentes métodos que se utilizan para pasar o transmitir parámetros.

Los métodos más empleados para realizar el paso de parámetros son:

- *paso por valor*(también conocido por *parámetro valor*),
- *paso por referencia o dirección* (también conocido por *parámetro variable*),
- *paso por nombre,*
- *paso por resultado.*

Paso de Parámetros Por Valor

El paso por valor se utiliza en muchos lenguajes de programación; por ejemplo, C, Modula-2, Pascal, Algol y Snobol. La razón de su popularidad es la analogía con los argumentos de una función, donde los valores se proporcionan en el orden de cálculo de resultados. Los parámetros se tratan como variables locales y los valores iníciales se proporcionan copiando los valores de los correspondientes argumentos.

Los parámetros formales, locales a la función, reciben como valores iníciales los valores de los parámetros actuales y con ello se ejecutan las acciones descritas en el subprograma.

No se hace diferente entre un argumento que es variable, constante o expresión, ya que sólo importa el valor del argumento. La Figura 4.13 muestra el mecanismo de paso por valores de un con un procedimientos con tres parámetros.

El mecanismo de paso se resume así:

Valor primer parámetro: A = 5.

Valor segundo parámetro: constante = 18.

Valor tercer parámetro: expresión B * 3 + 4 = 25.

Los valores 5, 18 y 25 se transforman en los parámetros X, Y, Z, respectivamente, cuando se ejecuta el procedimiento.

Aunque el *paso por valor* es sencillo, tiene una limitación acusada: *no existe ninguna otra conexión con los parámetros actuales,* y entonces los cambios que se produzcan por defecto del subprograma no produce cambios en los argumentos originales y, por consiguiente, no se pueden pasar valores de retorno al punto de llamada: es decir los *parámetros* son solo de *entrada.* El parámetro actual no puede modificarse por el subprograma. Cualquier cambio realizado en los parámetros formales durante la ejecución del subprograma se destruye cuando se termina el subprograma.

La *llamada por valor no devuelve información al programa que llama.*

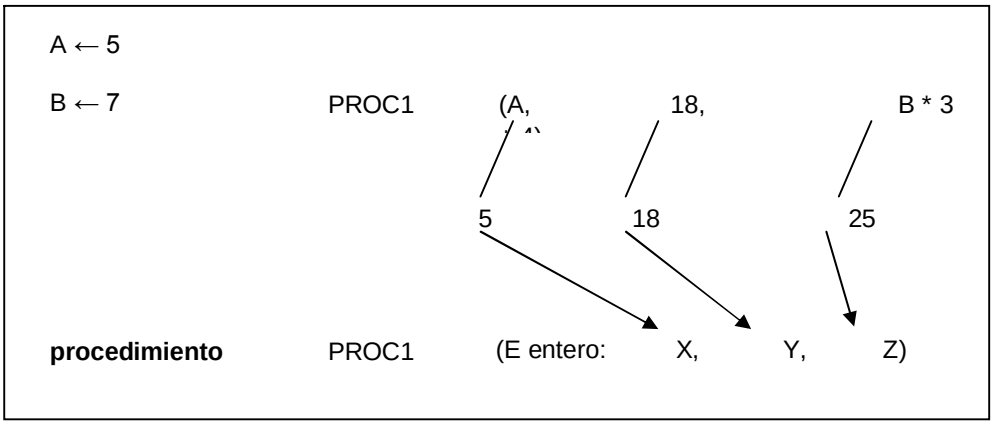


Figura 4.13 Paso de Parámetros Por Valor.

Existe una variante de la llamada por valor y es la llamada por *valor de resultado (*ver Figura 4.13*).* Las variables indicadas por los parámetros formales se inicializan en la llamada al subprograma por valor tras la ejecución del subprograma; los resultados (valores de los parámetros formales) se transfieren a los actuales. Este método se utiliza en algunas versiones de FORTRAN.

Paso de Parámetros Por Referencia

En numerosas ocasiones se requiere que ciertos parámetros sirvan como parámetros de salida, es decir, se devuelvan los resultados a la unidad o programas que llama. Este método se denomina *paso por referencia* o también de *llamada de dirección o variable.* La unidad que llama pasa a la unidad llamada la dirección del parámetro actual (que está en el ámbito de la unidad llamante). Una referencia al correspondiente parámetro formal se trata como una referencia a la posición de memoria, cuya dirección se ha pasado. Entonces una variable pasada como parámetro real es compartida, es decir, se puede modificar directamente por el subprograma.

Este método existe en FORTRAN, COBOL, Modula-2, Pascal, PL/1 y Algol 68. La característica de este método se debe a su simplicidad y su analogía directa con la idea de que las variables tienen una posición de memoria asignada desde la cual se puede obtener o actualizar sus valores.

El área de almacenamiento (direcciones de memoria) se utiliza para pasar información de entrada y/o salida; en ambas direcciones.

En este método los parámetros son de entrada/salida y los parámetros se denominan *parámetros variables.*

Los parámetros valor y parámetros variables se suelen definir en la cabecera del subprograma. En el caso de lenguajes como Pascal, los parámetros variables deben ir precedidos por la palabra clave **var;**

```
program muestra;

/ /parámetros actuales a y b, c y d paso por referencia

  procedure prueba (var x,y:integer);

  begin / / procedimiento

/ /proceso de los valores de x e y

  end;

begin

1. prueba (a, c) ;

2. prueba (b, d) ;

end;
```

La primera llamada en (1) produce que los parámetros *a* y *c* sean sustituidos por *x* e *y* si los valores de *x* e *y* se modificaran dentro de *a* y *c* en el algoritmo principal. De igual modo, *b* y *d* son sustituidos por *x* e *y*, y cualquier modificación de *x* o *y* en el procedimiento afectará también al programa principal.

La *llamada por referencia* es muy útil para programas donde se necesita la comunicación del valor en ambas direcciones.

Notas:

Ambos métodos de paso de parámetros se aplican tanto a la llamada de funciones como a las de procedimientos:

1. Una función tiene la posibilidad de devolver los valores al programa principal de dos formas: a) como valor de la función, b) por medio de argumentos gobernados por la llamada de referencia en la correspondencia parámetro actual-parámetro formal.

2. Un procedimiento sólo puede devolver valores por el método de devolución de resultados.

El lenguaje Pascal permite que el programador especifique el tipo de paso de parámetros y, en un mismo subprograma, unos parámetros se pueden especificar por valor y otros por referencia.

```
procedure Q (i:integer; var  j:integer);
begin
    i := i + 10;
    j := j + 10;
   write (i, j)
end;
```

Los parámetros formales son i, j donde I se pasa por el valor y j por referencia.

Comparaciones entre los Métodos de Paso de Parámetros

Para examinar de modo práctico los diferentes métodos, consideremos un ejemplo único y veamos los diferentes valores que toman los parámetros, El algoritmo correspondiente con un procedimiento SUBR:

```
algoritmo DEMO
var
    entero: A,  B, C
inicio    / / DEMO
     A ← 3
     B ← 10
     C ← 17
     llamar_a SUBR (A, A, A + B, C)
     escribir (C)
fin       / /DEMO

procedimiento SUBR (<Modo> entero: x, y; E entero:z; <Modo> entero: v)

inicio
   x ← x + 1
   y ← y + z
fin_procedimiento
```

Modo por Valor

a) sólo por valor:

no se transmite ningún resultado, por consiguiente,

C no varía C = 17

b) valor_resultado:

x = A = 3

A = 3 *y = A = 3*

B = 5 *pasa al procedimiento* *z = A + B = 8*

C =17 *v = C = 17*

al ejecutar el procedimiento quedará:

x = x + 1 = 3 + 1 = 4

v = y + z = 3 + 8 = 11

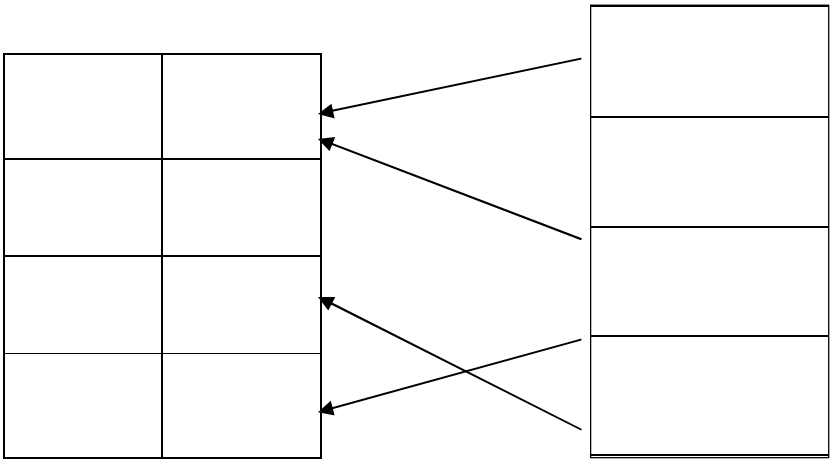
el parámetro llamado *v* pasa por el valor del resultado *v* a su parámetro actual correspondiente, C.

Por tanto, *C = 11*.

Modo por Referencia

Posiciones de la Memoria del Programa llamador

Posiciones de la Memoria del Subprograma llamado



C recibirá el valor *12.*

Utilizando variables globales

```
algoritmo DEMO
var entero: A,  B, C
inicio
    A ← 3
    B ← 5
    C ← 17
    llamar_a SUBR
    escribir (c)
fin
procedimiento SUBR
inicio
   a ← a + 1
   c ← a + a + b
fin_procedimiento
```

Es decir, el valor de C será 13.

La llamada por referencia es el sistema estándar utilizando por FORTRAN para pasar parámetros.

La llamada *por nombre* es estándar en Algol 60. Simula 67 proporciona llamadas por valor, referencia y *nombre.*

Pascal permite el paso por valor y por referencia.

```
procedure demo (y: integer; var z: real);
```

Especifica que se pasa por valor mientras que z se pasa por referencia, indicado por la palabra reservada **var***.* La elección entre un sistema u otro puede venir determinada por diversas consideraciones, como evitar efectos

laterales no deseados provocados por modificaciones inadvertidas de parámetros formales.

Síntesis de la Transmisión de Parámetros

Los métodos de transmisión de parámetros más utilizados son *por valor y por referencia.*

El paso de un parámetro por valor significa que el valor del argumento, *parámetro actual o real,* se asigna al parámetro formal. En otras palabras, antes de que aquel subprograma comience a ejecutarse, el argumento se evalúa a un valor específico (por ejemplo, 8 ó 12). Este valor se copia entonces en el correspondiente parámetro formal dentro del subprograma.

Una vez que el procedimiento arranca, cualquier cambio del valor de tal parámetro formal no se refleja en un cambio en el correspondiente argumento. Esto es, cuando el subprograma se termine, el argumento actual tendrá exactamente el mismo valor cuando el subprograma comenzó, con independencia de lo que ha sucedido al parámetro formal. Este método es el método por defecto en Pascal si no se indica explícitamente otro. Estos parámetros de entrada se denominan *parámetros valor*. En los algoritmos indicaremos como <modo> E (entrada). El paso de un *parámetro por referencia o dirección* se llama *parámetro variable,* en oposición al parámetro por valor. En este caso, la posición o dirección (no el valor) del argumento o parámetro actual se envía al subprograma. Si a un parámetro formal se le da el atributo de parámetro variable ─en Pascal con la palabra reservada **var**─ y si el parámetro actual es una variable, entonces un cambio en el parámetro formal se refleja en un cambio en el correspondiente parámetro actual, ya que ambos tienen la misma posición de memoria (ver Figura 4.14).

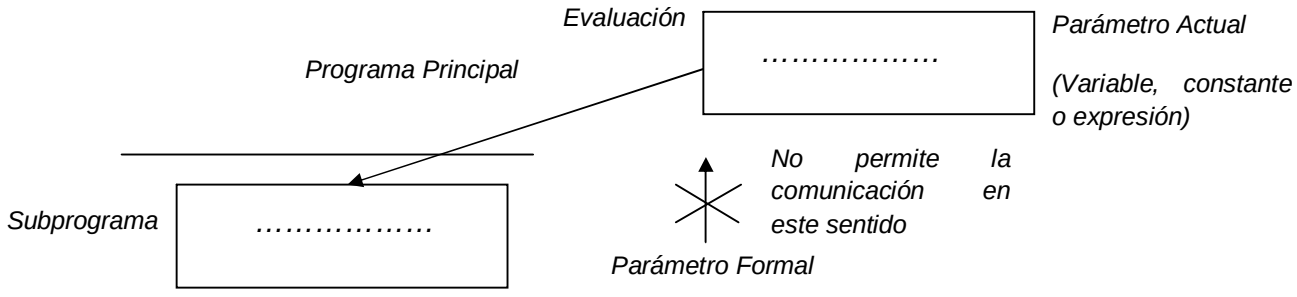


Figura 4.14 Paso de un Parámetro Por Valor.

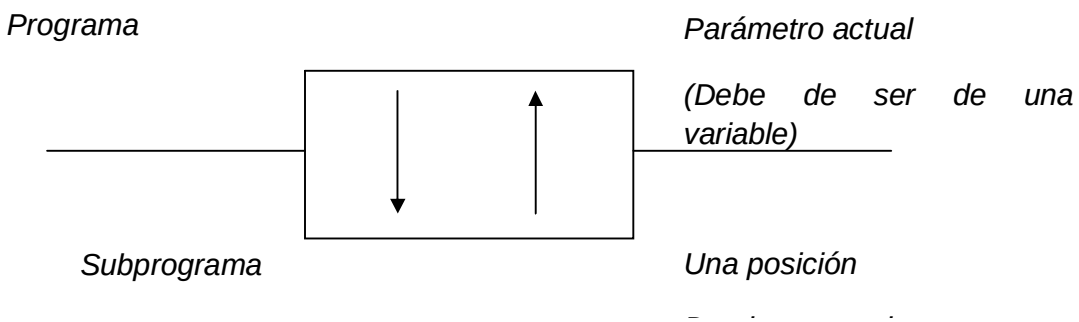


Figura 4.15 Paso de un Parámetro Por Referencia

Para indicar que deseamos transmitir un parámetro por dirección (ver Figura 4.15), lo indicaremos con la palabra *parámetro variable,* en Pascal se indica con la palabra reservada **var**, y especificaremos como <*modo*> E/S (*entrada/salida*) o S (*salida*).

Ejemplo:

Se trata de realizar el cálculo del área de un círculo y la longitud de la circunferencia en función del valor del radio leído desde el teclado.

Recordemos las fórmulas del área del círculo y de la longitud de la circunferencia:

A= pi x r^2 = pi x r x r

C= 2pi x r = 2 x pi x r *donde pi= 3.141592*

Los parámetros de entrada*: radio.*

Los parámetros de salida: *área, longitud*

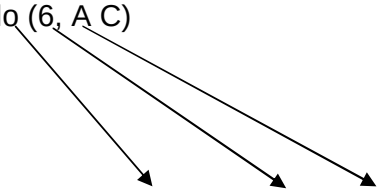
El procedimiento *circulo* calcula los valores pedidos.

```
procedimiento circulo  (E real: radio: area, longitud)
/ /parámetros valor: radio
/ /parámetros variable: area, longitud
  var
```

```
   real: pi
  inicio
     pi← 3.141592
     area← pi *radio *radio
     longitud ← 2 * pi * radio
  fin_procedimiento
```

Los parámetros formales son: radio, área, longitud, de los cuales son valores *(radio)* y variable *(área, longitud). Se in*voco el procedimiento *circulo* utilizando la instrucción:

```
  llamar_a circulo (6, A C)
   / / { programa principal
  inicio
  / /llamada al procedimiento
    llamar_a circulo (6, A C)
        .
        .
  fin
  procedimiento circulo (E real: radio; S real: area, longuitud)
  / /parametros valor: radio
  / /parametros variabler: area, longitud
      pi← 3.141592
  inicio
      area← pi *radio *radio
      longitud ← 2 * pi * radio
  fin_procedimiento
```



Ejemplo:

Considerando un subprograma N *con dos parámetros formales:* i, *transmitido por valor, y* j, *por variable.*

```
algoritmo M
 / / variables A, B enteras
 var
    entero: A, B
 inicio
     A ← 2
     B ← 3
      llamar_a N(A,B)
     escribir(A, B)
fin / / algoritmo M

procedmiento N(E entero: I; E/S entero: j)
     / /parámetro valor i
     / /parámetro variable j
inicio
     i ← i + 10
     j ← j + 10
     escribir (i, j)
fin_procedimiento
```

Si se ejecuta el procedimiento *N*, veamos qué resultados se escribirán:

A y B son parámetros actuales.

I y j son parámetros formales.

Como *i* es por valor, se transmite el valor de *A* a *i,* es decir, *i = A = 2.* Cuando *i* se modifica por defecto de *i← i + 10 a 12, A* no cambia y, por consiguiente, a la terminación de *N, A* sigue valiendo *2*.

El parámetro *B* se transmite por referencia, es decir, *j* es un parámetro variable. Al comenzar la ejecución de *N, B* se almacena como el valor *j* y cuando se suma *10* al valor de *j,i* en sí mismo no cambia. El valor del parámetro *B* se cambia a *13*. Cuando los valores *i,j* se escribe en *N*, los resultados son:

12 y 13

pero cuando retorna a *M* y al imprimir los valores de *A y B,* sólo ha cambiado el valor *B.* El valor de *i= 12* se pierde en N cuando éste ya termina. El valor de *j* también se pierde, pero éste es la dirección, no el valor *13*.

Se escribirá como resultado final de la instrucción escribir *(A, B):*

2 13

Funciones y Procedimientos como Parámetros

Hasta ahora los subprogramas que hemos considerado implican dos tipos de parámetros formales: *parámetros valor* y *parámetros variable.* Sin embargo en ocasiones se requiere que un procedimiento o función dado invoque a otro procedimiento o función que ha sido definido a fuera del ámbito de ese procedimiento o función, Por ejemplo, se puede necesitar que un procedimiento P invoque a una función F que puede estar o no definida en el procedimiento P; esto puede conseguirse transfiriendo como parámetros el procedimiento o la función externa (F) o procedimiento o función dado (por ejemplo, P). En resumen algunos lenguajes de programación ─entre ellos Pascal─ admiten *parámetros procedimientos y parámetros función.*

Ejemplos:

procedimiento P(E func: F1; E real: x,y)

real **función** F(E func; F1,F2; E entero:x,y)

Los parámetros formales del procedimiento P son la función F1 y las variables X e Y, y los parámetros formales de la función F son las funciones F1 y F2, y las variables x e y.

Parámetros en una Función

Para ilustrar el uso de los parámetros en una función emplearemos la función *integral* para calcular el área bajo una curva f(x) para un intervalo a <= x <= b.

La técnica conocida para el cálculo del área es subdividir la región en rectángulos, como se muestra en la Figura 4.16 y sumar las áreas de los rectángulos. Estos rectángulos se construyen subdividiendo el intervalo [a, b] en m subintervalos iguales y formando rectángulos con estos subintervalos como base y alturas dadas por los valores de f en los puntos medios de los subintervalos.

La *función integral* debe tener los parámetros formales *a, b* y *n*, que son parámetros valor ordinario actuales de tipo real; se asocian con los parámetros formales *a* y *b*; un parámetro actual de tipo entero, las subdivisiones, se asocia con el parámetro formal *n* y una función actual se asocia con el parámetro formal *f*.

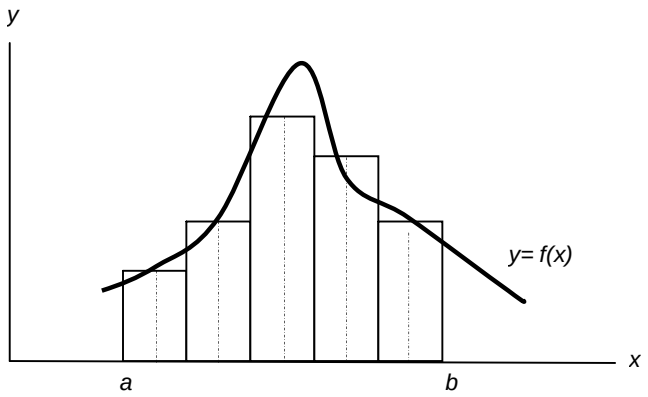


Figura 4.16 Cálculo del área bajo la curva f(x).

Los parámetros función se designan como tales como una cabecera de función dentro de la lista de parámetros formales. La función integral podrá definirse por:

real **función** integral (E func: f; E real: a,b; E entero: n)

el tipo func-tipo real función (E real:x) : func-

aquí la función func: F especifica que f es una función parámetro que denota una función cuyo parámetro formal y valor son de tipo real. El correspondiente parámetro función actual debe ser una función que tiene una parámetro formal real y un valor real, Por ejemplo, si integrado es una función de valor real con un parámetro de tipo real función es decir, de tipo :func

Area←Integral(integrado, 0, 1.5, 20)

es una referencia válida a función

Diseñar un algoritmo que utilice la función integral para calcular el área bajo es gráfico de la función es f1 $(x)=x^3-6x^2+10x$ *e integrado por integrado* $(x)=x^2+3x+2$ *para* $0<=x<=4$.

algoritmo Area_bajo_curvas

tipo

real **función** (E real: x) : func

var

real : a,b

entero : n

inicio

escribir (‘¿Entre que limites?’)

leer(a,b)

escribir (‘¿Subintervalos?’)

leer(n)

escribir (integral (f1, a, b, n))

escribir (integral (integrado, a, b, n))

fin

real **función** (E real: x) : func

inicio

devolver (x * x + 3 * x + 2)

```
    r ← n
fin_funcion
real función integral (E func: f; E real: a,b; E entero: n)
var
    real    : baserectangulo,altura,x,s
    entero : i
inicio
    baserectangulo ← (b-a)/ n
    x ← a + baserectangulo/2
    s ← 0
    desde i ← 1 hasta n hacer
altura ← f(x)
        s ← s + baserectangulo * altura
        x ←x + baserectangulo
    fin_desde
    devolver(s)
fin_funcion
```

Los Efectos Laterales

Las modificaciones que se produzcan mediante una función o procedimiento en los elementos situados fuera del subprograma (función o procedimiento) se denominan *efectos laterales*. Aunque en algunos casos los efectos laterales pueden ser beneficiosos en la programación, es conveniente no recurrir a ellos de modo general. Consideramos a continuación los efectos laterales en funciones y en procedimientos.

Los efectos laterales en procedimientos.- La *comunicación* del procedimiento con el resto del programa se debe realizar normalmente a través de los parámetros. Cualquier otra *comunicación* entre el procedimiento y el resto del programa se conoce como *efectos laterales.* Como ya se ha

comentado, los efectos laterales son perjudiciales en la mayoría de los casos, como se indica en la Figura 4.17.

Si un procedimiento modifica una variable global (distinta de un parámetro actual), este es un *efecto lateral*. Por ello, excepto en contadas ocasiones, no debe aparecer en la declaración de procedimiento. Si se necesita una variable temporal en un procedimiento, utiliza una variable local, no una variable global. Si se desea que el programa modifique el valor de una variable global, utilice un parámetro formal variable en la declaración de procedimiento y a continuación utilice la variable global como el parámetro actual en una llamada en el procedimiento.

En general, se debe de conseguir la regla de <<*ninguna variable global en procedimiento*>>, aunque esta prohibición no significa que los procedimientos no puedan manipular variables globales. De hecho, el cambio de variables globales se debe de pasar al procedimiento como parámetros actuales. Las variables globales no se deben utilizar directamente en las instrucciones en el cuerpo de un procedimiento; en su lugar, utilice un parámetro formal o variable local.

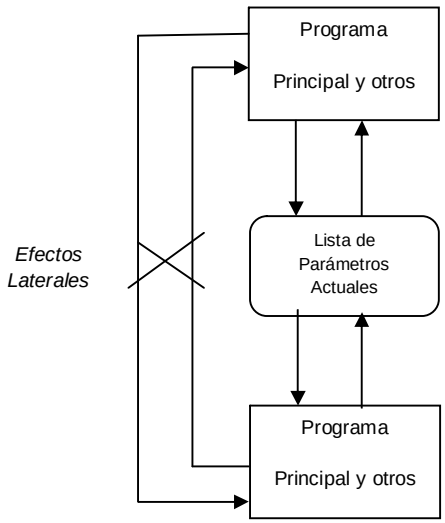


Figura 4.17 Efectos Laterales en Procedimientos.

En aquellos lenguajes en que es posible declarar constantes, como Pascal, se pueden utilizar constantes globales en una declaración de procedimiento; la razón reside en el hecho de que las constantes no pueden ser modificadas por

el procedimiento y, por consiguiente, no existe peligro de que se puedan modificar inadvertidamente.

Los efectos laterales en funciones. Una función toma los valores de los argumentos y devuelve un *único valor*. Sin embargo, al igual que los procedimientos, una función, en algunos lenguajes de programación, puede hacer cosas similares a un procedimiento o subrutina. Una función puede tener parámetros variables además de parámetros valor en la lista de parámetros formales. Una función puede cambiar el contenido de una variable global y ejecutar instrucciones de entrada/salida (escribir un mensaje en la pantalla, leer un valor en el teclado, etc.). Estas operaciones se conocen como *parámetros laterales* y se deben evitar.

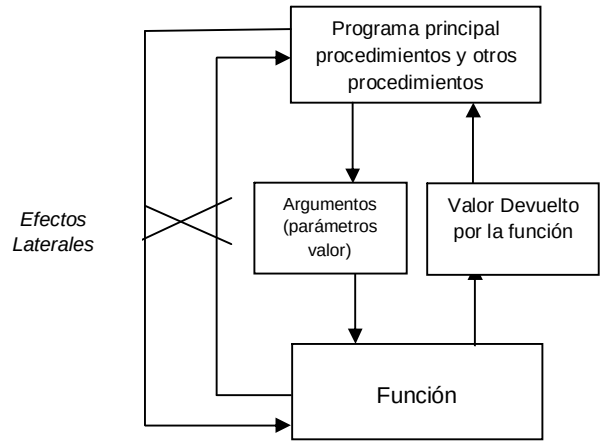


Figura 4.18 Efectos Laterales de una Función.

Los efectos laterales (ver Figura 4.18) están considerados normalmente como una mala técnica de programación, pues hacen más difícil de entender los programas.

Toda la información que se transfiere entre procedimientos y funciones debe realizarse a través de la lista de parámetros y no a través de variables globales. Esto convertirá el procedimiento o función en módulos

independientes que pueden ser comprobados y depurados por sí solos, lo que evitará no preocuparnos por el resto de las partes del programa.

Ejemplo.- Este algoritmo lee un número decimal entero positivo desde teclado y lo transforma el número a notación romana. El número leído deberá ser menor de 3000.

Algoritmo sin programación modular

```
algoritmo Romanos
  var entero : n,digito,r,j
inicio
  repetir
    escribir ('Deme numero')
    leer(n)
    hasta_que (n >= 0) Y (n <= 3000)
    r ← n
    digito ← r div 1000
    r ← r mod 1000
    desde j ← 1 hasta digito hacer
       escribir('M')
    fin_desde
    digito ← r div 100
    r ← r mod 100
    si digito = 9 entonces
       escribir('C', 'M')
    si_no
    si digito > 4 entonces
       escribir('D')
       desde j ← 1 hasta digito – 5 hacer
```

```
        escribir ('C')
      fin_desde
    si_no
      si digito = 4 entonces
      escribir('C', 'D')
      si_no
          desde j ← 1 hasta digito hacer
           escribir('C')
          fin_desde
      fin_si
    fin_si
fin_si
digito ← r div 10
r ← r mod 10
si digito = 9 entonces
  escribir('X', 'C')
si_no
   si digito > 4 entonces
     escribir('L')
      desde j ← i hasta digito – 5 hacer
          escribir('X')
          fin_desde
  si_no
      si digito = 4 entonces
          escribir('X', 'L')
      si_no
```

```
            desde j ← 1 hasta digito hacer
               escribir('X')
            fin_desde
         fin_si
      fin_si
   fin_si
   digito ← r
   si digito = 9 entonces
    escribir('I', 'X')
   si_no
    si digito > 4 entonces
      escribir ('V')
    desde j ← 1 hasta digito – 5 hacer
      escribir('I')
    fin_desde
  si_no
    si digito = 4 entonces
      escribir('I', 'V')
    si_no
      desde j ← 1 hasta digito hacer
          escribir('I')
       fin_desde
       fin_si
    fin_si
   fin_si
fin
```

Mediante Programación Modular

```
algoritmo Romanos
var n,r,digito: entero
inicio
  repetir
  escribir('Deme numero')
  leer(n)
  hasta_que  (n >= 0) Y (n <= 3000)
  r ← n
  digito ← r div 1000
  r ← r mod 1000
  calccifrarom(digito, 'M', ' ', ' ')
  digito ← r div 100
  r ← r mod 100
  calccifrarom(digito, 'C', 'D', 'M')
  digito ← r div 10
  r ← r mod 10
  calccifrarom(digito, 'X', 'L', 'C')
  digito ← r
  calccifrarom(digito, 'I', 'V', 'X')
fin

procedimiento calccifrarom(E entero:digito; E carácter:v1,v2,v3)
var entero: j
inicio
```

```
    si digito = 9 entonces
        escribir(v1, v3)
      si_no
        si digito > 4 entonces
          escribir(v2)
           desde j ← i hasta digito – 5 hacer
              escribir(v1)
              fin_desde
        si_no
          si digito = 4 entonces
              escribir(v1, v2)
          si_no
              desde j ← 1 hasta digito hacer
                 escribir(v1)
              fin_desde
          fin_si
        fin_si
      fin_si
fin_procedimiento
```

Capítulo 5 “Diseño del Sistema Inteligente de Aprendizaje (ProgEst)”

En este capítulo se abordar el diseño del Sistema Inteligente de Aprendizaje (ProgEst), aplicado al dominio de programación estructurada. A través del grafo conceptual, se muestra la relación que existe entre cada concepto o habilidad y el objetivo instruccional (OI). Lo anterior para instrumentar el plan instruccional en términos de las actividades que deben realizar tanto el sistema de aprendizaje inteligente como el alumno.

5.1 Los Modelos Cognitivos

El objetivo de un modelo cognitivo es desarrollar una efectiva simulación de la solución del problema en un determinado dominio desde el punto de vista del humano. En esta técnica el conocimiento se divide en componentes que guardan una relación directa con la forma en que el humano los clasifica y los utiliza.

El mérito de esta aproximación es que proporciona un módulo experto cuya taxonomía permite un proceso tutorial y una comunicación con el estudiante más profunda. Aunque ha habido avances importantes en los últimos años, por parte de las ciencias cognitivas, estos modelos requieren gran cantidad de tiempo en su desarrollo; por el gran número de detalles que tienen que ser incorporados.

En esta técnica se tienen que considerar básicamente tres cuestiones: qué componentes procedentes del análisis cognitivo son importantes para el proceso tutorial, con qué nivel deben ser representados los componentes, y finalmente cómo deben ser tratados los diferentes tipos de conocimiento: procedimental, declarativo y cualitativo en esta técnica de modelado.

5.2 Organización del Domino de Enseñanza "Programación Estructurada"

De acuerdo a Estévez mencionado en Laureano-Cruces, Terán-Gilmore, De Arriaga & El Alami, (2003), abordamos problemas profundos cuando nos preguntamos: 1) ¿los estudiantes están realmente aprendiendo? 2) ¿qué grado de aplicación tiene el conocimiento adquirido dentro del entorno para el cual se están preparando?; 3) ¿son suficientes los conocimientos para continuar una educación superior?; y 4) debido al ritmo de desarrollo científico y tecnológico que conlleva la participación en el conocimiento de varias disciplinas, ¿cuándo queda obsoleto lo aprendido?

Ante tal perspectiva pretendemos enriquecer el diseño didáctico con las ciencias cognitivas con el fin de incluir procesos de este tipo en la enseñanza y el aprendizaje mediante el diseño y el uso de estrategias cognitivas. Así, la propuesta de estos innovadores modelos se sustenta en: 1) una perspectiva multinodal (basada en diferentes fuentes teóricas), 2) un enfoque holístico e integral (conocimientos, habilidades, actitudes y valores); 3) privilegiar el empleo de estrategias cognitivas como medio para activar los procesos mentales necesarios para propiciar el aprendizaje; 4) utilización de los componentes cognitivos de la habilidad a adquirir incluyendo los propios modelos mentales del experto.

5.2.1 Diseño Didáctico

Este diseño se refiere a la consideración del "currículo", definido en términos del proyecto global en el que está incluida la actividad concreta. Y en el caso de temas específicos se refiere a lo que todo docente debe conocer sobre su material de enseñanza para propiciar su adquisición.

Así el diseño didáctico se concibe como un cuerpo de conocimientos que se ocupa de: 1) la comprensión, el mejoramiento y la aplicación de métodos de enseñanza, 2) la combinación óptima de métodos, y 3) los contextos o situaciones en los que se espera que dichos métodos de enseñanza produzcan mejores resultados.

Al resultado de un diseño didáctico se le conoce como modelo y es éste la representación final de un conjunto integrado por componentes estratégicos

que nos permitirá: 1) secuenciar el material, 2) utilizar grafos conceptuales, 3) utilizar ejemplos, 4) la incorporación de la práctica en un determinado momento, y 5) el uso de estrategias para motivar a los estudiantes. Otro aspecto importante de este modelo didáctico es que debe mostrar los diferentes aspectos que entran en juego en la enseñanza, con el fin de alcanzar los objetivos deseados del mejor modo posible y bajo las condiciones anticipadas. En la Figura 5.1, se muestra el grafo genético que se utilizará para organizar el material didáctico propuesto en Laureano-Cruces, et al., (2008a).

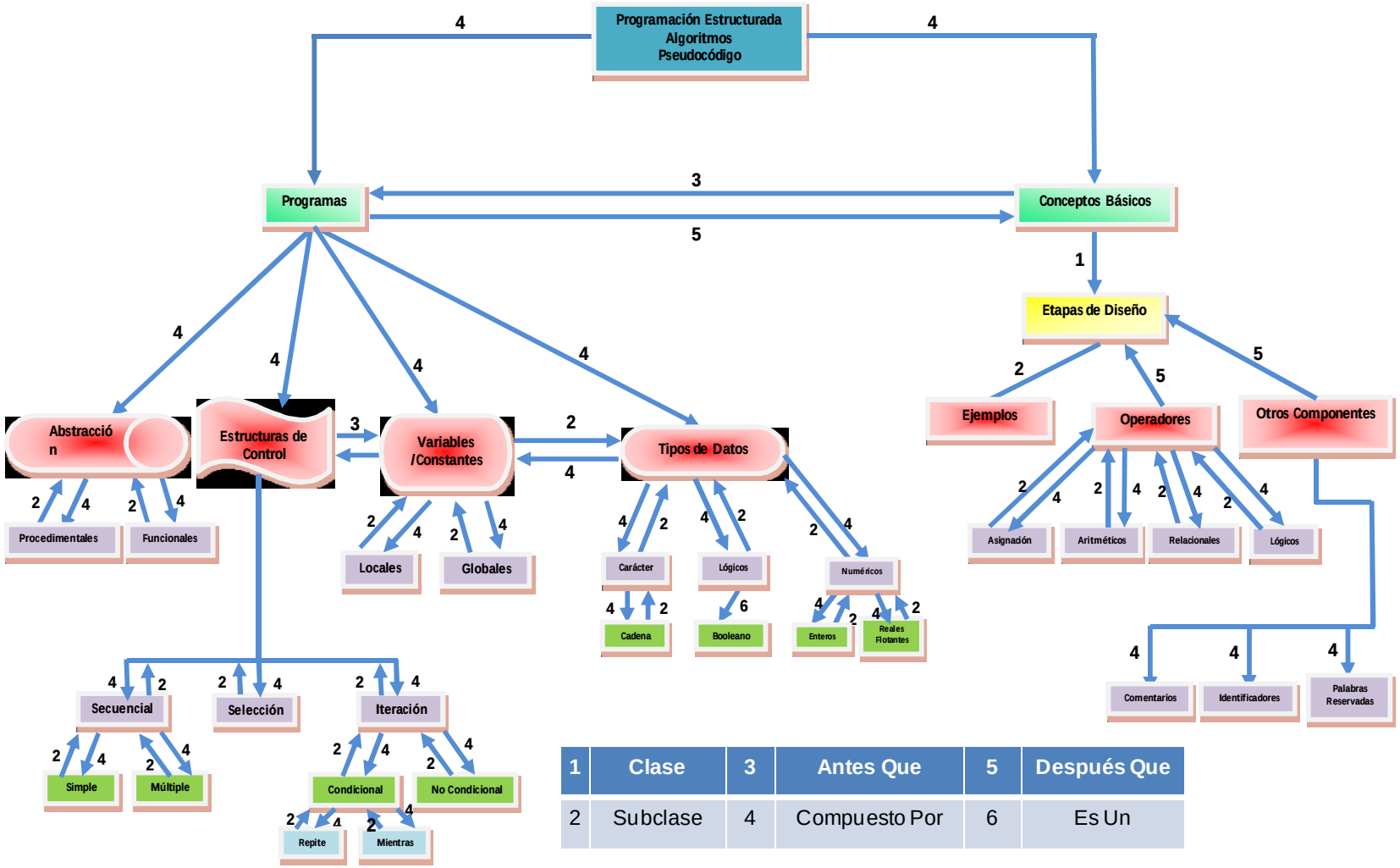
De acuerdo a la propuesta de Laureano-Cruces, De Arriaga & El Alami, (2004b), el Grafo Genético (GG), es una herramienta para representar el conocimiento que nació de la inspiración de Goldstein, (1976) y Goldstein, (1979), basada en la epistemología genética de Piaget mencionada en Ginsburg & Opper, (1986). De forma general el GG muestra el conocimiento (de cualquier tipo) agrupado en islas y enlaces para relacionarlas. Estos enlaces pueden ser de orden o de inclusión. También se puede registrar la historia y el estilo de aprendizaje de un estudiante, tomando en cuenta las islas visitadas y la predilección de enlaces utilizados en el recorrido del estudiante durante el aprendizaje.

Los enlaces que se pueden mostrar en este GG son y pueden ser ampliados según las necesidades del dominio a modelar, en este caso nos basaremos en el ejemplo de enlaces propuesto en Laureano-Cruces, (1997), Fernández, (1989) y Ramírez-Rodríguez, (1994):

1. PreCond.- Implica un orden de precedencia **antes que**.
2. PostCond.- Conlleva un orden de posterioridad, un conocimiento al que se puede acceder **después que** se cubra el conocimiento al que se está enlazando.
3. Anlg: analogía.- Cuando dos islas están enlazadas de esta manera, existe una correspondencia de las constantes de una a las de la otra isla.
4. Comp: componente.- Este enlace implica que un conocimiento o habilidad está **compuesto por** otro componente.
5. Clase.- Supone la existencia de una jerarquía conceptual o de habilidades.
6. Subclase.- Exige la existencia de niveles de granularidad en la definición de abstracciones conceptuales o de habilidades.
7. Es Un.- Representa la definición de un componente específico de acuerdo con el dominio en cuestión.

Los enlaces entre las islas, además de indicar el orden de ejecución de la tarea, explicitan las relaciones y los datos de entrada y salidas que habrá entre las islas y los diferentes niveles de abstracción (pueden o no existir) que representan la ejecución del SAI.

El dominio “Programación Estructurada con algoritmos en pseudocódigo” se describe en el capítulo 4. El Grafo Genético está representado en la Figura 5.1.



1	Clase	3	Antes Que	5	Después Que
2	Subclase	4	Compuesto Por	6	Es Un

Figura 5.1 Grafo Genético

5.3 Objetivos Instruccionales

En esta sección, se muestra la relación que existe entre cada concepto o habilidad y el objetivo instrucional (OI). Donde un OI está definido por las habilidades y capacidades cognoscitivas que el tutor desea transmitir al alumno. Los OI pueden ser clasificados de acuerdo a la siguiente taxonomía Bloom, (1956): 1) conocimiento, 2) comprensión, 3) aplicación, 4) análisis, 5) síntesis, 6) evaluación.

Conocimiento: este objetivo está relacionado con la adquisición del conocimiento por parte del alumno.

El alumno debe conocer las definiciones de los conceptos teóricos divididos en:

1) Tipos de Datos.

2) Estructuras de Control (Secuencia, Iteración-Condicional, No-Condicional).

3) Selección–Simple y Múltiple).

4) Abstracciones (Procedurales y Funcionales).

5) Tipos De Parámetros (Referencia y Valor).

6) Variables.

7) Constantes.

Comprensión: debe manejar la idea de la lógica detallada de cada abstracción que realice. Ser capaz de entender las estructuras de control y su significado, con el fin de trasladar el concepto de cada una de ellas, a nuevos contextos, que le permitan interpretar y comparar.

Aplicación: este objetivo está relacionado con la puesta en práctica de los conocimientos de las etapas anteriores.

De acuerdo a Laureano-Cruces, et al., (2003), y adaptado a nuestro contexto el alumno aplicará correctamente el procedimiento para ser capaz de:

1) Acotar el uso de los distintos tipos de datos de acuerdo a las características del problema.
2) La clase de estructura de control, *ad-hoc* al objetivo del módulo que se pretende diseñar.

3) Comprender el uso de los distintos tipos de abstracción que nos proporciona el paradigma de la programación estructurada.
    a. Paso de parámetros y sus tipos.
    b. Tipos de abstracción: funcional y procedural.

Análisis: servirá para enseñar al alumno a comprender los estados de los distintos escenarios, y analizarlos en función de distintos tipos de razonamiento propuestos en Laureano-Cruces, Terán-Gilmore & De Arriaga, (2004a) que conlleven:

Predicción: acción o efecto de anunciar lo que sucederá a partir de información incompleta en un posible futuro.

Post-dicción: explicación de cómo se ha llegado a una situación actual concreta, equivale a una predicción retroactiva por ello es considerada como un tipo de razonamiento no deductivo también conocido como lógica abductiva. Este tipo de razonamiento consta de dos fases: la *primera*, consiste en un conjunto de explicaciones posibles y la *segunda* en la construcción de explicaciones y la selección de la mejor.

Interpretación cuantitativa: dada la descripción parcial de las particularidades de una situación y algunas observaciones de su comportamiento, inferir qué otras particularidades existen y qué más puede suceder.

Razonamiento causal: causa-efecto. Este tipo de razonamiento es una herramienta para dar crédito a una hipótesis que proviene de un comportamiento observado o postulado. Es útil para la generación de: explicaciones, interpretación de medidas, planificación de experimentos y obviamente aprendizaje.

El alumno analizará un determinado estado del escenario y deberá ser capaz de saber qué pasa con base en los valores de los distintos parámetros y sus relaciones, con los demás elementos.

Una vez que se han determinado los objetivos instruccionales, se debe refinar el plan instruccional en términos de las actividades que deben realizar tanto el sistema de aprendizaje inteligente como el alumno. Estas actividades las denominaremos *estrategias instruccionales* y se encargan de: proponer al alumno ejercicios, motivar al alumno, hacer llegar al alumno las comunicaciones del sistema (mediante explicaciones, comentarios, una muestra gráfica, etc.) y dar una continuidad a la sesión instruccional.

Los escenarios se construyeron con base en el grafo genético del dominio "*Programación Estructurada con Algoritmos en Pseudocódigo*" (ver Figura 5.1) y los objetivos instruccionales propuestos en Laureano-Cruces, et al., (2008a).

Cada escenario cuenta con los parámetros de evaluación que se definen en la siguiente sección (5.3.1). Estas medidas son necesarias para determinar: qué explicar, qué nivel de detalle, en qué momento y la manera de interrumpir al alumno. Es un proceso que también incluye algunos aspectos afectivo-motivacionales (estrategias operativas), con el fin de mantener y controlar la ejecución continúa de tareas y actividades necesarias en el estudio.

Una vez que los objetivos instruccionales (OI) se han determinado, éstos deben ser refinados en un plan instruccional de las actividades que deben llevarse a cabo por el Sistema de Aprendizaje Inteligente y el estudiante. Estas actividades se denominan estrategias instruccionales y deben: 1) proponer una serie de ejercicios para el estudiante, 2) motivar al estudiante, 3) hacer que el estudiante interactúe con el Sistema a través de explicaciones, comentarios, figuras de ejemplo, entre otras, 4) dar continuidad en la sesión instruccional. La continuidad es administrada por el SAI, asimismo sus intervenciones serán a través de la interfaz.

Actualmente existen diferentes tipos de estrategias instruccionales que pueden ser utilizadas en un proceso de enseñanza-aprendizaje. Éstas se dividen en dos grupos: 1) Estrategias Cognitivas y 2) Estrategias Operativas.

Estrategias Cognitivas.- determinan las actividades en las que el tutor debe involucrarse con el estudiante (dar una explicación, hacer un ejercicio práctico, realizar un examen, entre otros) con el fin de lograr la aplicación del objetivo instruccional (OI).

Estrategias Operativas.- permitirán al tutor conducir la clase, orientar, contextualizar y motivar (por medio de estrategias afectivas) al estudiante todo el tiempo.

En la siguiente sección (5.3.1) se describe la clasificación para el tipo de estrategias que se pueden aplicar, a fin de mejorar y hacer amigable la interacción con los estudiantes.

Estrategias Cognitivas (acciones relativas al diagnóstico cognitivo)

Estrategias para la presentación del material instruccional:

- Dar 1) un concepto general ó 2) resumido ó 3) detallar (paso a paso) la explicación del concepto.
- Mostrar ejemplos de funcionamiento por expertos.
- Proponer un ejercicio práctico de: 1) un procedimiento de operación, 2) análisis de un error.

Estrategias para evaluar los conocimientos y habilidades adquiridos por el estudiante:

- Evaluar con un examen.
- Evaluar con un ejercicio práctico

Estrategias para el tratamiento de errores:

- Estrategias para la descripción del error.
- Explicación de los errores: desde un nivel superficial a un nivel detallado.

Estrategias para la corrección del error:

- Informar: explicación de los conocimientos.
- Consolidar: explicación del comportamiento.
- Ignorar.

Estrategias para corregir el error:

- Rehacer el ejercicio de la práctica.

- Mostrar la solución.
- Proponer un ejercicio similar.
- No hacer nada.

Estrategias Operativas

Estrategias de ubicación (contextualizar al estudiante):

- Presentar la sesión al estudiante.
- Fin de la sesión.
- Informar al estudiante: 1) la(s) acción (es) siguiente(s) a desempeñar, 2) las acciones iniciales a desempeñar, 3) la(s) última(s) acción(es) a realizar.
- Para ir a través de: 1) un concepto, 2) de la sesión previa, 3) los prerrequisitos de un determinado concepto.

Estrategias de motivación de estudio (en el estudiante):

- Felicitación al estudiante.
- Animar al estudiante.

Estrategias para orientar o guiar al estudiante en su desempeño:

- Asesorar.
- Prevenir.
- Plantear una pregunta.
- Ayuda.

Estrategias para captar la atención del estudiante:

- Captar la atención del estudiante.

De acuerdo a Ames mencionado en Laureano-Cruces, Mora-Torres, De Arriaga, Ramírez-Rodríguez & Escarela-Pérez, (2008b) y citado por Jaques, (2004) el aprendizaje en los estudiantes es motivado por un objetivo interno o externo.

El conjunto de estrategias operativas es instrumentar una interacción amigable, y uno de sus objetivos es mantener el interés del estudiante por un tiempo más largo. Estas intervenciones se aplicarán en el texto que se muestra al estudiante a través de la interfaz.

Los estudiantes con objetivo de aprendizaje interno, se orientan al desarrollo de nuevas habilidades y capacidades a alcanzar, a tratar de mejorar su nivel de competencia y aprender cosas nuevas. Tienen su propia motivación de estudio, completan algunas de las actividades por el placer que ofrece esta sensación. Estas personas hacen un mayor esfuerzo cuando se enteran de algo nuevo o se enfrentan a un desafío. Son conocidos como estudiantes motivados internamente.

Los estudiantes con objetivo de aprendizaje externo desean expresar la importancia de su compromiso hacia su obligación o tarea, demostrar que su obligación es importante, y que tienen habilidades. Se sienten exitosos cuando complacen al profesor con un mejor desempeño que el resto de los estudiantes. Cuando la experiencia de aprendizaje se dificulta, no les gusta aumentar el esfuerzo, porque eso demostraría que no tienen la habilidad suficiente (es decir, se sienten vulnerables, bajo su punto de vista). Estos estudiantes están motivados por factores externos (valores, los padres, acreditar el curso). Son conocidos como estudiantes motivados externamente.

Con esto en mente y basados en las obras de Jacques & Viccari, (2004a), Jacques, (2004b), Gutiérrez, (1994), y Laureano-Cruces, et al., (2008b) se aplicó un cuestionario creado con base en Pintrich, Smith, García y Mckeachie, (1991) para determinar el tipo de orientación de la motivación de estudio del alumno, de igual forma cabe mencionar que el cuestionario fue utilizado y descrito por Jacques, et al., (2004a). Para este caso el cuestionario apoyará al sistema ProgEst a diagnosticar en el estudiante el tipo de motivación de estudio (interno o externo).

5.3.1. Elementos que conforman el Modelo del Proceso de Enseñanza-Aprendizaje

Los elementos del modelo del proceso de Enseñanza Aprendizaje están conformados por los nueve elementos del tutor de didáctica general propuesto

en Laureano-Cruces, et al., (2008b), más el estilo de aprendizaje y la motivación de estudio (sea éste interno o externo). Quedando integrados de la siguiente forma: 1) interés, 2) deseo, 3) ayuda, 4) estrategias cognitivas y operativas, 5) interrupción, 6) renuncia, 7) aprendizaje, 8) tiempos inactivos, 9) error, 10) estilo de aprendizaje, 11) motivación: interna o externa.

En las secciones siguientes se combinan los elementos que son evaluados por los distintos agentes y su relación con los errores manejados en el caso de estudio.

5.4 Modelo del Estudiante

Entre los elementos que se deben considerar para evaluar el desempeño de un maestro humano, así como para determinar la forma en que el *módulo tutor* del SAI funciona, están la motivación por parte del estudiante, los materiales disponibles, los objetivos instruccionales y las habilidades del estudiante. Sánchez-Guerrero, et al., (2009)

Por otro lado, el diseño de un SAI debe ser lo suficientemente flexible como para adaptarse a diferentes necesidades y diferentes alumnos, así como proporcionar un mecanismo que haga que el estudiante esté consciente de sus mejoras. Por lo tanto, los nueve elementos en el cuadro de la Tabla 5.1, son esenciales para simular el proceso de enseñanza-aprendizaje en un SAI.

Tabla 5.1. Elementos considerados en el proceso de enseñanza-aprendizaje.

Elementos	Descripción	Relacionados con
Interés	Interés en el tema de la tarea propuesta.	Motivación, perfil.
Deseo	Deseo de continuar realizando la tarea propuesta.	Motivación, perfil.
Ayuda	Posibilidad de pedir ayuda para llevar a cabo la tarea propuesta.	Confianza en el profesor y el medio ambiente.
Estrategias Cognitivas/ Operativas	Contar con estrategias cog/op.	Motivación del profesor.
Interrupción	Necesidad de interrumpir la tarea propuesta.	Capacidad para organizar, el uso cognoscitivo y / o estrategias operativas, conducir un debate constructivo, pedir tiempo.
Renuncia	Posibilidad de salir del sistema sin terminar la tarea propuesta	La posibilidad latente del estudiante que no tiene interés y / o conocimientos previos.
Aprendizaje	Cumplir con la tarea propuesta, aprende a realizar la tarea propuesta al desarrollar las habilidades y las competencias necesarias.	Posibilidad latente a la que siempre se desea llegar, y se alienta con base en las diferentes estrategias.
Tiempos inactivos	Posibilidad de no realizar ninguna tarea durante períodos prolongados.	Falta de comprensión o de interés en el ejemplo o concepto, por el agotamiento o la frustración.
Errores	Posibles errores en el desempeño de la tarea propuesta	Nivel no ha sido alcanzado en el manejo de las diferentes competencias; por la distracción, la falta de interés, el agotamiento.

La interpretación de cada uno de los elementos involucrados en el proceso de enseñanza-aprendizaje y su relación con los demás elementos se detalla en Laureano-Cruces, et al., (2008a).

Con base en los datos anteriores se elaborarán las distintas estrategias de enseñanza-aprendizaje. Éstas están relacionadas con los distintos tipos de errores manejados por diferentes agentes. Dichos agentes representan micromundos de experticia en alguno de los objetivos instruccionales.

5.5 Diseño de los Agentes

Contamos con un modelo mental que, de forma genérica, implica el uso de las distintas estructuras de control. Este punto es de suma importancia, y se encuentra en la etapa de la lógica detallada de cada uno de los módulos (programación modular descendente y programación estructurada). Considerando esto y con base en los objetivos instruccionales (definidos en la sección 5.3), contamos con tres agentes:

1) Tipos de Datos, Variables y Constantes.

2) Estructuras de Control:
   - Secuencia.
   - Iteración: condicional (*mientras* y *repite*) y no-condicional (*progresión aritmética*).
   - Selección: sencilla (*si_entonces_sino*) y múltiple (*caso*).

3) Abstracciones:
   - Procedimientos.
   - Funciones.

Estos agentes están representados por la Figura 5.2.

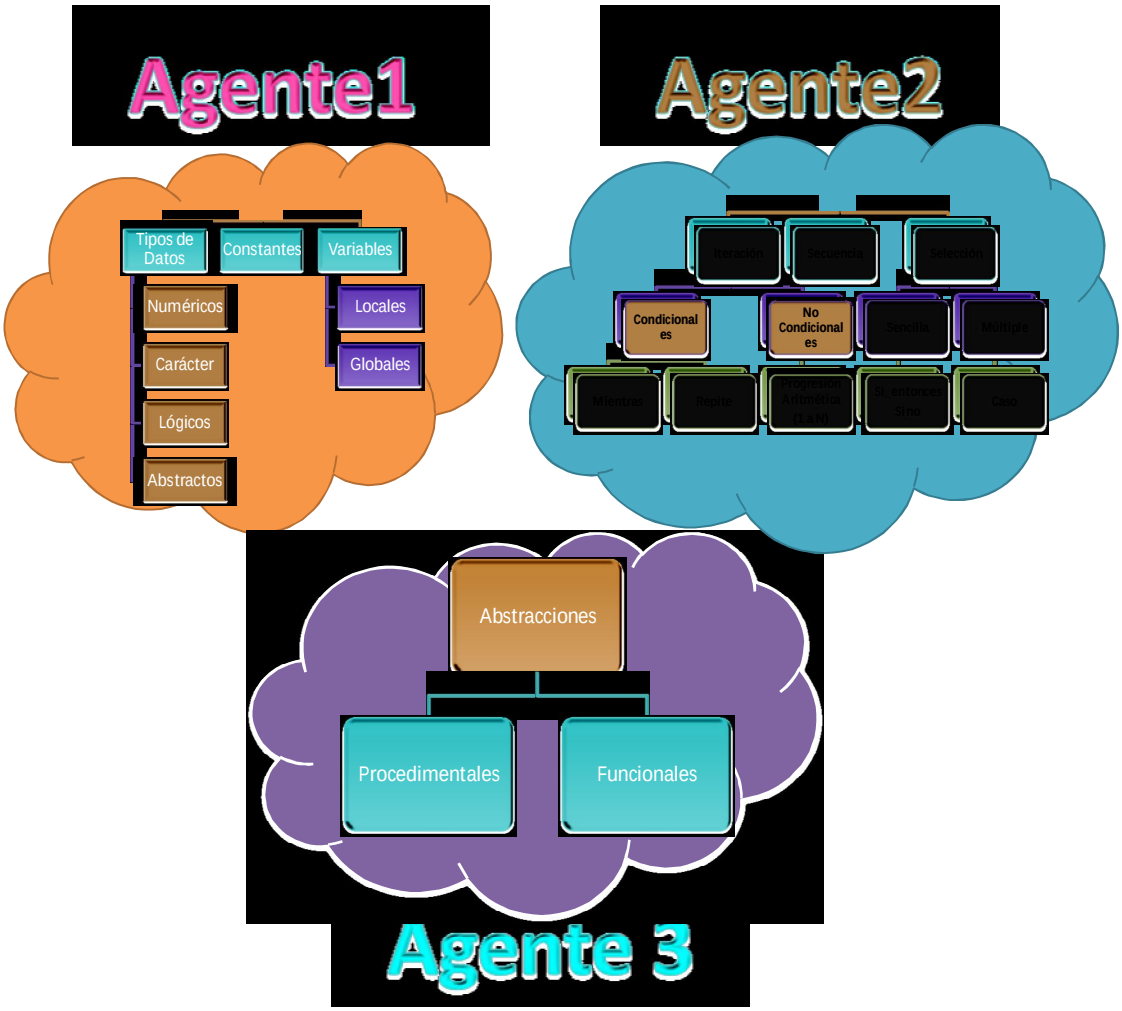


Figura 5.2 Dominio de cada uno de los Agentes.

El proceso de enseñanza-aprendizaje está basado en el Grafo Genético anterior (ver Figura 5.1); cuyos nodos en el más alto nivel de abstracción representan los objetivos instruccionales. Siguiendo la arquitectura propuesta por Laureano-Cruces, et al., (2000a), estos objetivos quedan inmersos en una arquitectura multiagente.

5.5.1 Modelo MultiAgente

Como lo describe Laureano-Cruces, et al., (2000a), la arquitectura MultiAgente y su anatomía está compuesta de tres partes:

- Una presentación (P).
- Una abstracción (A).
- Un control de dialogo (CD).

La *presentación* es la parte del agente que es vista por el mundo; está relacionada con alguna técnica de presentación.

5.2.1 Diseño Didáctico

La abstracción representa el estado local del agente; es la parte donde se encuentran los objetos conceptuales para poder tener acceso al dominio; en ella se encuentra implementada la competencia del agente.

En cuanto a la parte llamada *controlador de diálogo,* es donde se lleva a cabo la coordinación entre la abstracción y la presentación y la coordinación con otros agentes en caso de ser necesaria; cabe mencionar que pueden existir agentes sólo con la parte de presentación o sólo con la parte de abstracción o con ninguna de éstas.

Cada pieza de la abstracción de un agente consta de un componente que representa la *interfaz con el corazón funcional (ICF)*; ésta será el puente entre lo que el agente debe hacer con el dominio (los objetos conceptuales) y el *corazón funcional (CF).* El CF está constituido por los elementos del dominio. Por otro lado en la contraparte; *la presentación* consta de un componente que contiene las *técnicas utilizadas en la presentación (CTP)* del sistema hacia el exterior, y éste a su vez consta

de un componente que será el receptor directo de los *elementos en el más bajo nivel de abstracción (CIBN),* esto es: mouse, teclado, sensores, motores, ruedas, etc.

En ProgEst tenemos tres agentes independientes: 1) "*Tipos de Datos, Variables y Constantes*", 2) "*Estructuras de Control*" y 3) "*Abstracciones*" (ver Figura 5.2), la tarea que desarrollan es de forma jerárquica, con lo que se resuelve el problema de intervención durante el desarrollo de la sesión. Los agentes revisarán la tarea en determinados puntos críticos (ubicados por el experto) y lo harán de forma ordenada debido a la jerarquía mencionada. En el caso de que alguno de ellos encuentre una falla entrará en acción su mecanismo interno.

Como lo mencionamos en el capítulo dos el comportamiento de los agentes está dividido en dos subagentes; el primero es el *diagnóstico* y su objetivo es prestar atención al entorno (cómo avanza el estudiante). Con la información obtenida se adquiere evidencia de las capacidades perceptuales del agente. Para saber si el estudiante emplea, no emplea, o emplea de forma incorrecta, la habilidad monitorizada y controlada de forma exclusiva por ese agente. Partiendo de sus indagaciones detecta el o los errores cometidos y entonces entra en acción el segundo subagente, representado por un *MicroMundo* con la misión de crear un entorno que ayude al estudiante a aclarar sus dudas, todo a través de las tácticas didácticas que guiarán su intervención hasta el final del proceso y el alumno sea restablecido en el entorno principal, donde incurrió en el error.

En ProgEst la intervención del *MicroMundo* consiste en presentar una explicación del tema seguida de ejemplos completos que muestran el uso correcto de la habilidad.

ProgEst, cuenta con la capacidad de solicitar "ayuda" dentro del escenario, permitiendo mostrar el contenido del tema respectivo, antes de ser invocado el *MicroMundo*. Esto debido al enfoque de entrenador de nuestro SAI.

En su nivel de abstracción, los tres agentes de ProgEst, "*Tipos de Datos, Variables y Constantes*", "*Estructuras de Control*" y "*Abstracciones*", están representados por las Figuras, 5.3, 5.4 y 5.5 respectivamente. En dichas figuras, la abstracción está constituida por los errores clasificados de acuerdo al agente (ver Tablas 5.3, 5.4 y 5.5) y las tácticas didácticas (ver Tablas 5.6 y 5.7). Por el otro lado, la presentación está simbolizada por los medidores y actuadores que son los mecanismos de percepción a través de la interfaz como: 1) la presentación del OA de acuerdo al estilo de aprendizaje, 2) todos los mecanismos de intervención tutorial, tales como: ejemplos, texto, etc., y 3) las intervenciones de las estrategias operativas de acuerdo al tipo de motivación de estudio, entre otras.

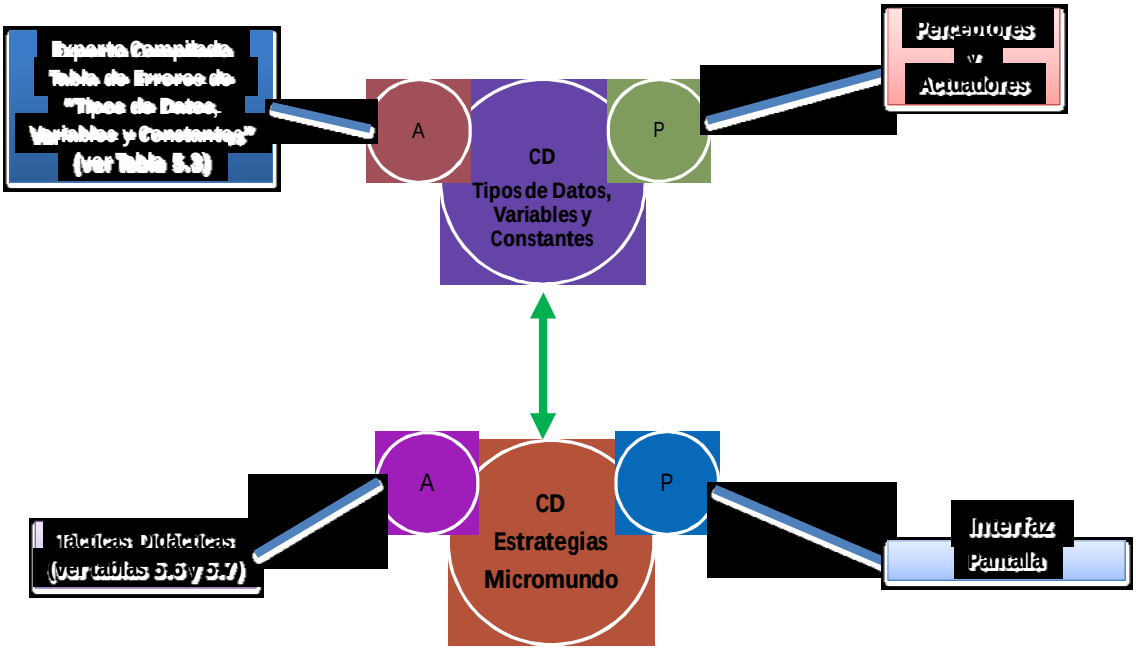


Figura 5.3 Representación del Agente "Tipos de Datos, Variables y Constantes"

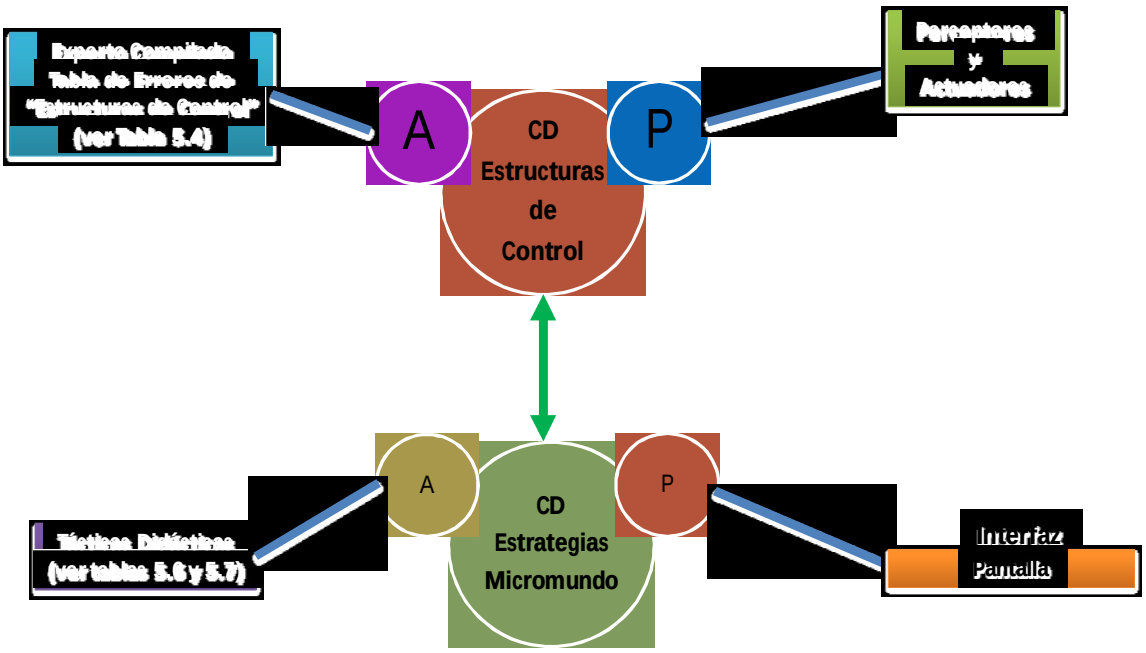


Figura 5.4 Representación del Agente "Estructuras de Control".

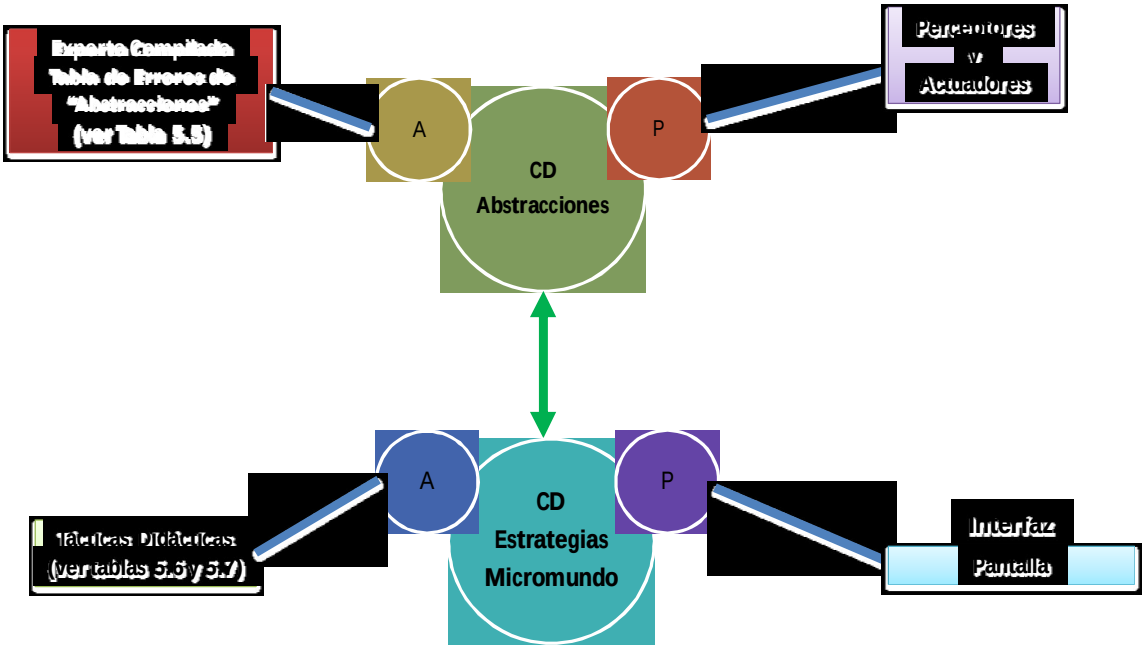


Figura 5.5 Representación del Agente "Abstracciones".

El modelo funcional de estos agentes, se encuentra representado por las Figuras 5.6, 5.7, 5.8 y 5.9, donde el funcionamiento global de un agente visto como un subtutor, se encuentra dividido en dos subagentes que representan la PercepciónDeUnAgente y la AcciónDeUnAgente visualizados en las figuras antes mencionadas.

Percepción de un Agente

En las Figuras 5.6, 5.7 y 5.8 se muestra la parte relativa a la percepción. En ella se encuentran los atributos que son capaces de percibir de acuerdo a sus capacidades y en un instante dado, donde el entorno en curso se encontrará en un determinado estado. A nivel de acciones, lo anterior se traduce en la revisión, que por parte de ese agente, tendrá lugar en el desarrollo de la tarea cognitiva del alumno. En los agentes, las capacidades de percepción están representadas por los errores que le competen a él. En caso de error, este agente lo detectará y se lo comunicará al agente *MicroMundo*.

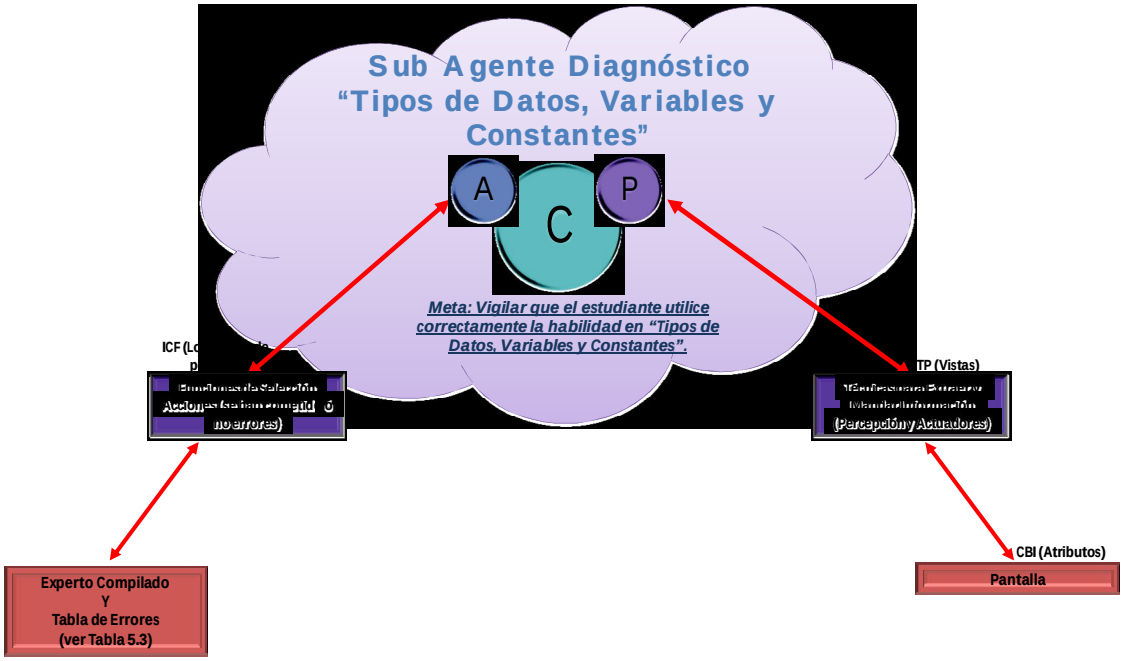


Figura 5.6 Representación del Sub Agente Diagnóstico de "Tipos de Datos, Variables y Constantes".

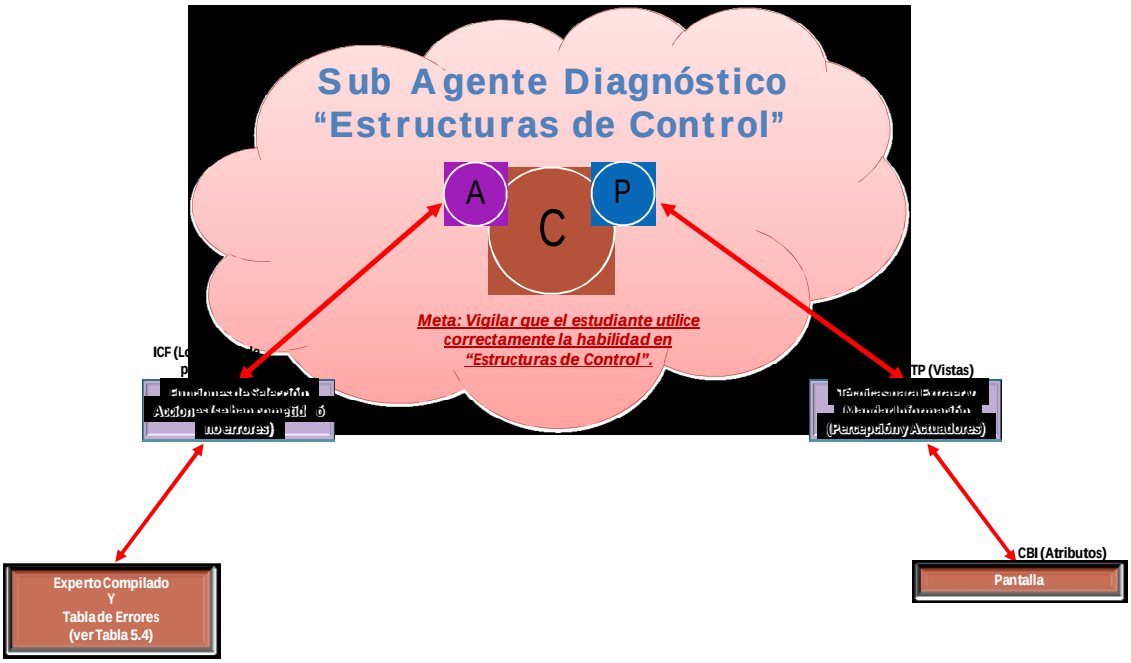


Figura 5.7 Representación del Sub Agente Diagnóstico de "Estructuras de Control"

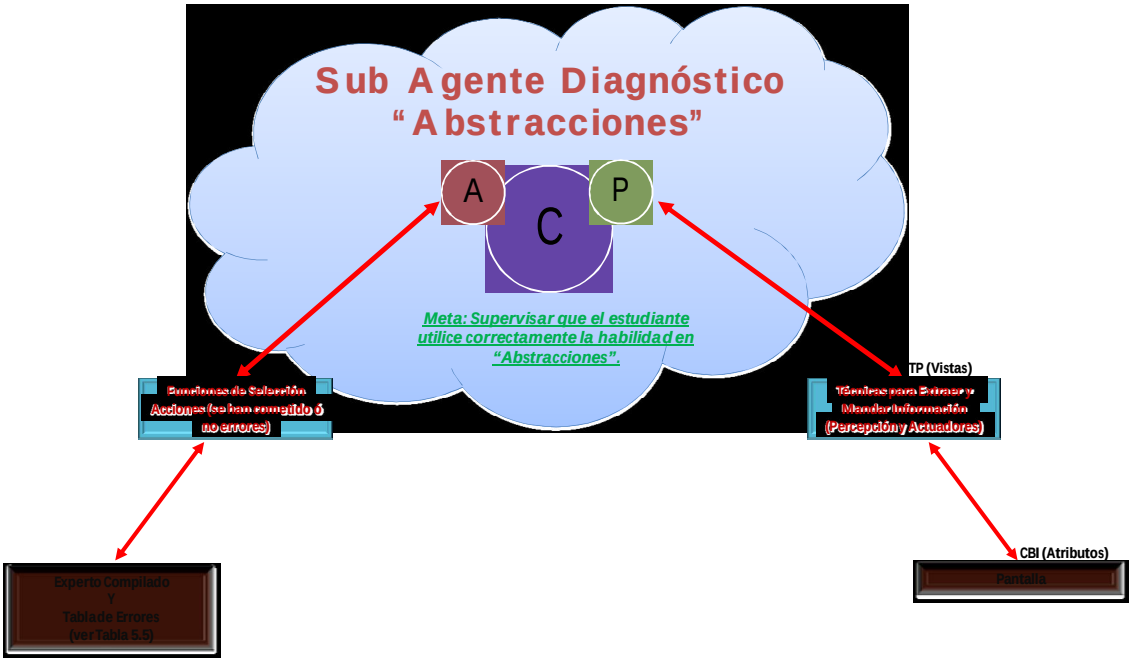


Figura 5.8 Representación del Sub Agente Diagnóstico de "Abstracciones".

La Acción de un Agente

En la Figura 5.9 se muestra la parte relativa a la acción de dicho agente, lo que se traduce en que de acuerdo a la meta, que es tratar el error producido por el estudiante se elegirá una táctica correctiva la cual conlleva una serie de acciones didácticas que tendrán efecto en el entorno representado por la pantalla.

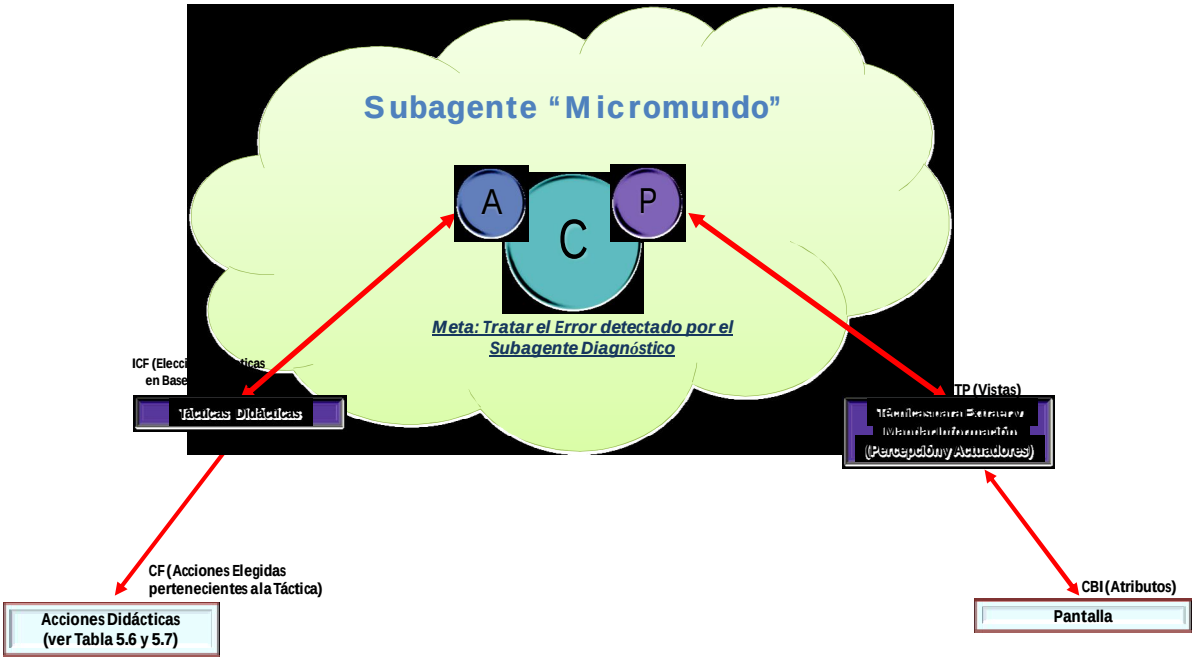


Figura 5.9 Representación del Sub Agente "Micromundo".

Anatomía del Agente (Subtutor)

De acuerdo a lo anterior tenemos dividido el funcionamiento de un agente (subtutor), en dos subagentes, dónde la meta del Agente *MicroMundo* es impuesta por el Agente *Diagnóstico*. La habilidad que manejan los diferentes subtutores se encuentra en la Tabla 5.2.

Sub Agente Diagnóstico

Meta: Manejar la habilidad de forma correcta y en el orden indicado de prioridad.
Clase de Meta: *Activa.*

- ***Percepción de un Agente (Diagnóstico)***

Acciones que Percibe: Revisar a través de sus perceptores o sea, la pantalla.
Lo que puede Percibir: Errores que le competen.
Lo que Percibirá: Errores cometidos en un *instante dado*.

- ***La Acción de un Agente (Diagnóstico)***

Acciones de un Agente: Activar al agente *MicroMundo* en caso de error, en caso contrario continuar con la labor de centinela durante el desarrollo.

Sub Agente *MicroMundo*

Meta: Tratar el error detectado a través de una táctica didáctica correctiva.
Clase de Meta: *Pasiva.*

- ***Percepción de unAgente (MicroMundo)***

Acciones que Percibe: Expone a través de sus actuadores la táctica didáctica correctiva involucrada con el tipo de error.
Lo que puede Percibir: Conocer los resultados del MiniTest, practicado al estudiante.
Lo que Percibirá: Conocer si el estudiante ha comprendido, o no, el concepto erróneo a través de los resultados del MiniTest.

- ***La Acción de un Agente (MicroMundo)***

Acciones de un Agente: Regresar al entorno original una vez tratado el error y en caso de cometer un error fatal se saca del MiniTest y se envía al objeto de aprendizaje establecido con base al estilo de aprendizaje representado por diferentes colores (obtenidos del cuestionario CHAEA), que contiene el material de estudio con las recomendaciones pertinentes.

Tabla 5.2 Habilidades que revisan los subtutores.

Agente	Habilidad
SubTutor 1	Tipos de Datos, Variables y Constantes
SubTutor 2	Estructuras de Control
SubTutor 3	Abstracciones

De esta forma se implementó el circuito de interacción del sistema compuesto por tres agentes y el entorno. A través de la interfaz se guardan los datos del seguimiento del desarrollo del estudiante.

5.6 Los Errores

Para los errores se desarrolló una clasificación basada en la experiencia de autora de este trabajo y de otros expertos en el dominio. De esta manera, los errores se clasifican en: graves, leves y fatales. Basados en la arquitectura multiagente con intervención dinámica, propuesta por Laureano-Cruces, et al., (2000b) y Laureano-Cruces, et al., (2000a). Se hace una clasificación en "*especialistas*", éstos, son agentes reactivos que entran en acción en el instante en el que se comete un error, que compete a su experticia. En la Tabla 5.3 se muestra la clasificación para el Agente 1 "*Tipos de datos, Variables y Constantes*", en la Tabla 5.4 para el Agente 2 "*Estructuras de Control*" y en la Tabla 5.5 para el Agente 3 "*Abstracciones*".

Los ***errores graves (G)***, implican una falta importante de conceptualización, lo que conlleva al fracaso de la aplicación en el diseño y construcción de los algoritmos en pseudocódigo.

Los ***errores leves (L)***, implican falta de atención global, más que falta de conocimiento, esto es, cuentan con el conocimiento específico, incluso pueden haberlo utilizado previamente, sin embargo, por alguna falta de atención se desorientan y no completan parte del proceso.

Los ***errores fatales (F)***, implican una falta total de conocimiento, con lo cual se considera imposible proseguir, recordemos que este tutor es de entrenamiento y se presupone un conocimiento inicial del método de resolución, visto en clase.

Tabla 5.3 Clasificación de Errores para controlar la intervención del SUB-TUTOR de "Tipos de Datos, Variables y Constantes" (obtenidos a partir del desarrollo global de la TC).

Errores	Tipos de Datos
Error 1	No detecta su uso del tipo de datos F
Error 2	No detecta su uso de las variables G
Error 3	No detecta su uso de constantes G
Error 4	No detecta el área específica para definir las constantes L
Error 5	No detecta el uso del tipo de la constante G
Error 6	No detecta el uso de los diferentes tipos de variables F
Error 7	No detecta del uso de tipos datos de los numéricos F
Error 8	No detecta el uso de las variables de tipo entero L
Error 9	No detecta el uso de las constantes de tipo enteras L
Error 10	No detecta el uso de las variables de tipo flotante L
Error 11	No detecta el uso de las constantes de tipo flotantes L
Error 12	Confunde el uso de las variables de tipo entero y flotantes G
Error 13	No detecta el uso del tipo carácter F
Error 14	No detecta el uso de las variables de tipo carácter F
Error 15	No detecta el uso de las constantes de tipo carácter G
Error 16	No detecta el uso de las cadenas de caracteres F
Error 17	No detecta el uso de las variables de tipo cadenas de carácter F
Error 18	No detecta el uso de las variables lógicas F
Error 19	No detecta el uso de las variables booleanas G

Tabla 5.4 Clasificación de Errores para controlar la intervención del SUB-TUTOR de "Estructuras de Control" (obtenidos a partir del desarrollo global de la TC).

Errores	Estructuras de Control
Error 1	No detecta el uso de las estructuras de control **G**
Error 2	No detecta el uso estructura de control Secuencia **G**
Error 3	No detecta su uso de la estructura de control iteración **G**
Error 4	No detecta su uso de la estructura de control de selección **G**
Error 5	Confunde las estructuras de control **G**
Error 6	Confunde la estructura de control de selección con la estructura de iteración **G**
Error 7	No sabe en qué caso se aplican estructuras de control iteración y selección **G**
Error 8	No sabe usar la condición de la estructura **G**
Error 9	No sabe utilizar el alcance (inicio y fin) de la estructura **G**
Error 10	No define correctamente el tipo de las variables de las estructuras **L**
Error 11	No utiliza correctamente el inicio de la estructura de control **L**
Error 12	No sabe distinguir entre preguntar por la condición al inicio o al final de la instrucción **L**
Error 13	No utiliza correctamente el incremento o decremento en la estructura de control de iteración **L**
Error 14	No define correctamente el uso de los operadores en la condición de la estructura **F**

Tabla 5.5 Clasificación de Errores para controlar la intervención del SUB-TUTOR de "Abstracciones" (obtenidos a partir del desarrollo global de la TC).

Errores	Abstracciones
Error 1	No detecta su uso de las abstracciones y es necesario aplicarlas F
Error 2	No detecta su uso de los procedimientos G
Error 3	No detecta su uso de las funciones G
Error 4	Confunde el uso de funciones y procedimiento F
Error 5	No detecta el uso de parámetros F
Error 6	No detecta el uso de paso de parámetros en un procedimiento F
Error 7	No detecta el uso de paso de parámetros en una función F
Error 8	No definir correctamente el tipo de la función L
Error 9	No definir correctamente el tipo del procedimiento L
Error 10	No usa en forma correcta el tipo en los parámetros del procedimiento G
Error 11	No usa en forma correcta el tipo de los parámetros en una función G
Error 12	No usa en forma correcta el regreso de los parámetros al programa principal o a donde fue invocado el procedimiento G
Error 13	No usa en forma correcta el regreso de los parámetros al programa principal o a donde fue invocado el procedimiento G
Error 14	No usa en forma correcta el lugar de definición de las funciones en el programa G
Error 15	No detecta el uso de las variables globales L
Error 16	No detecta el uso de las variables locales en una función L
Error 17	No detecta el uso de las variables locales en un procedimiento L
Error 18	No usa en forma correcta las variables globales y/o locales G
Error 19	No define en forma correcta el tipo de las variables locales del procedimiento L
Error 20	No define en forma correcta el tipo de las variables locales de la función L
Error 21	No usa en forma correcta (inicio y fin) del procedimiento G
Error 22	No usa en forma correcta (inicio y fin) de la función G
Error 23	No usa en forma correcta el paso de parámetros por valor en una función G
Error 24	No usa en forma correcta el paso de parámetros por valor en el procedimiento G
Error 25	No usa en forma correcta el paso de parámetros por valor en una función G
Error 26	No usa en forma correcta el paso de parámetros por referencia en el procedimiento G
Error 27	No usa en forma correcta el paso de parámetros por referencia en la función G

5.7 Estrategias para los Tipos de Errores

En las siguientes Tablas 5.6 y 5.7 se muestran las diferentes tácticas didácticas y las acciones didácticas a tratar por el subagente diagnostico, que se aplican en el ProgEst. Estas tablas se basan en el trabajo de Laureano-Cruces, et al., (2008b).

Tabla 5.6 Clasificación de las Tácticas Didácticas y las Acciones Didácticas a tratar por el subagente diagnostico, que se aplican en el ProgEst.

Evento	Objetivo	Expresiones para tácticas Cognitivas/Operativas, interrupciones y para pedir ayuda.	Acciones	Número de Oportunidades de Respuesta
Interés (1) Deseo de continuar (2)	Interna y Externa	• ¡Felicitaciones por el esfuerzo realizado, sigue así! • ¡Tú eres muy inteligente y estás obteniendo muy buen progreso en esta tarea!		
Pedir Ayuda (3)	Interno	• ¡Buen momento para expresar dudas! Este tema no es del todo sencillo.	• Se muestra la expresión. Posteriormente el sistema envía al alumno a consultar el material del tema y permite que una vez que se consulta la ayuda se puede continuar resolviendo el ejercicio.	• Las peticiones de ayuda no están condicionadas mientras se esté dentro del ejercicio.
	Externo	• ¡Hey! ¡Pedir ayuda no significa que no seas capaz de hacerlo!	• Se muestra la expresión. Posteriormente el sistema envía al alumno a consultar el material del tema y permite que una vez que se consulta la ayuda se puede continuar resolviendo el ejercicio.	• Las peticiones de ayuda no están condicionadas mientras se esté dentro del ejercicio.

Interrupción (5)	Interna	• ¿Te gustaría saber más de esto?	• Al momento de seleccionar el botón de Interrupción, se interrumpe la ejecución del ejercicio y se muestra la expresión.	• Las peticiones interrupción no e condicionadas mientra esté dentro del ejercicio
	Externa	• ¿Quieres más consejos?	• Al momento de seleccionar el botón de Interrupción, se interrumpe la ejecución del ejercicio y se muestra la expresión descrita en la columna anterior. • Se permite que una vez que se consulte a la ayuda se puede continuar resolviendo el ejercicio.	• Las peticiones interrupción no e condicionadas mientra esté dentro del ejercicio
Renuncia (6)	Interna y externa	• ¡Sigue intentando! ¡El éxito está en este camino!	• Al momento de seleccionar el botón de Salida, se termina la ejecución del ejercicio y se muestra la expresión.	• Al seleccionar el botó sistema ya no le permi alumno regresar al ejer y lo envía al mat escogido con base cuestionario CHAEA.
Aprendizaje (7)	Interna	• ¡Eres un triunfador! ¡Recuerda todos tus logros!	• Al resolver correctamente el ejercicio al alumno se despliega y se muestra la expresión.	
	Externa	• El éxito en la realización de esta tarea muestra las nuevas habilidades adquiridas.	• Al resolver correctamente el ejercicio al alumno se despliega y se muestra la expresión.	
Tiempo Inactivo (8)	Interna y externa	• ¡Hey! Es tiempo de empezar a trabajar	• Después de pasar 30 segundos se despliega y se muestra la expresión. • Si el alumno no responde al segundo tiempo inactivo (de 30 segundos) pierde una oportunidad de las 3 permitidas por el sistema para resolver el ejercicio.	• El máximo número tiempos de ocio qu alumno puede sol son 3.

Tabla 5.7 Clasificación de los Errores con base a las Tácticas Didácticas y las Acciones Didácticas a tratar por el subagente diagnostico que se aplican en el ProgEst.

Error (9)	Objetivo Interno /Externo	Número de Oportunidades de Respuesta	Acciones	Estrategia Operativa/ Cognoscitiva
Leve	Interno y Externo	Tres oportunidades	Le permite regresar al escenario y continuar, siempre y cuando el número de oportunidad no se ha agotado.	Le muestra la siguiente frase: ” Podemos lograrlo juntos. ¡Es importante seguir intentándolo!
Grave	Interno y Externo	Una oportunidad	Hace que reinicie el trabajo escenario.	Le muestra la siguiente frase: ” ¡Continuemos. Lo lograrás!
Fatal	Interno y Externo	Una oportunidad	Lo saca del escenario.	Le muestra la siguiente frase: ” Sigue intentando el éxito está en camino” y lo manda al material del dominio en el color de acuerdo al estilo de aprendizaje.

5.8 Escenarios del Caso de Estudio

Al evaluar un escenario se determina el valor de cada uno de los once elementos que intervienen en el proceso de enseñanza-aprendizaje (Sección 5.3), la detección de cada uno de los elementos puede ser de forma directa o indirecta. Como ejemplos de la primera está una petición de ayuda por parte del estudiante, el estilo u objetivo de aprendizaje a través de un cuestionario. En el caso de la forma indirecta, ésta se realiza por la inferencia que se hace a través de las relaciones de causalidad. Los escenarios que se muestran a continuación son representativos del proceso de enseñanza-aprendizaje del dominio específico. En este ejemplo se pretende evaluar los tipos de operadores de asignación, aritméticos, relacionales y lógicos.

Escenario 1:

Objetivo: El alumno analizará y aplicará los conceptos de las estructuras de control básicas (*secuencia, selección (bifurcación condicional) e iteración (cíclicos)*), en el diseño de algoritmos.

Instrucciones del Ejercicio:

- Analice el siguiente algoritmo y relaciónelo con las instrucciones de la Tabla 5.8, en los espacios en blanco escriba la respuesta correcta, con el fin de completar el algoritmo.

Algoritmo suma_numeros

/*Este algoritmo calcula los números múltiplos de tres y calcula la suma la suma los y los números que no cumplan con esta característica los eleva al cuadrado y los suma, esta serie se calcula para un intervalo dado desde teclado*/

Principal

Inicio

/*Declaración de los tipos de variables */

_____ inicial, final, cuadrado, suma_numeros, suma_multiplos

suma_numeros ← 0

suma_multiplos ←0

/*las variables son inicializadas con cero*/

_____(“Dame el intervalo inicial”);

/*aquí se lee los valor inicial*/

lee(_____)

escribe(“Dame el valor final del intervalo”)

lee(final)

/*Aquí inicia la estructura de ciclo donde se toma el valor inicial y se compara con el valor final si es menor o igual, entonces continua el ciclo*/

___(inicial < = _____)

_____

multiplo_tres ← _____ modulo 3;

si (_____ = 0)

inicio

_____ ← suma_multiplos + inicial;

escribe(“la suma de los múltiplos hasta este

momento es: “ ,_____);

fin

_____

Inicio

Cuadrado ← _____:

suma_numeros ← suma_numeros + _____ ;

escribe(“la suma hasta este momento es: “,

suma_numeros);

fin

inicial ← _____ + 1;

_____

escribe (“El valor final de la suma total de los múltiplos de tres es:”, suma_multiplos, “en el Intevalo”, inicial, “al”, suma_multiplos);

_____ (El valor final de la suma de los valores del intevalo:”, inicial “al”,final,” es:”, suma_numeros);

fin

En este escenario es el alumno relaciona la tabla de las respuestas y del algoritmo anterior.

Tabla 5.8 Opciones del Ejercicio de Estructuras de Control.

1) inicial	2) si_no	3) lee	4) final
5) suma_multiplos	6) cuadrado	7) escribe	8) inicio
9) fin	10) algoritmo	11) mientras	12) Múltiplo

En este escenario, el alumno inicialmente presentó interés en la tarea y deseo de continuar, además solicitó ayuda sobre el tema, sin embargo se detectaron tiempos inactivos, y hubo un solo error, el cálculo de *g*. Lo anterior significa que, de acuerdo al proceso de enseñanza-aprendizaje y aplicando la matriz de causalidad, se tiene el siguiente vector de entrada (V_i) y el vector final (V_f):

Tabla 5.9 Matriz de Causalidad del Escenario 1 (9 Elementos del Tutor de Didáctica General).

	Interés	Deseo	Ayuda	Estrategia cog/op	Interrup-ción	Renuncia	Aprend.	Tiempos inactivos	Error
V_i	**1**	**1**	**1**	0	0	0	0	**1**	**1**
V_1	0.000045	0.000045	0.500000	1.000000	0.993307	0.993307	0.993307	0.500000	0.500000
V_2	0.000572	0.000572	0.993078	1.000000	0.924110	0.924110	0.000591	0.993078	0.993078
V_3	0.000000	0.000000	0.999999	1.000000	0.999999	0.999999	0.000101	0.999999	0.999999
V_4	0.000000	0.000000	1.000000	1.000000	1.000000	1.000000	0.000045	1.000000	1.000000
V_5	0.000000	0.000000	1.000000	1.000000	1.000000	1.000000	0.000045	1.000000	1.000000
V_f	0.000000	0.000000	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.000045	**1.000000**	**1.000000**

El vector final indica que incluso solicitando la ayuda para resolver el ejercicio, existe la posibilidad de que el alumno continúe presentando errores y tiempos inactivos. Esto genera la posibilidad de renuncia. De lo anterior se infiere que pierde el interés y el deseo de continuar, además de no presentarse el aprendizaje. Por esta razón, se precisa urgentemente la interrupción por parte del tutor, con el fin de aplicar una estrategia cognitiva/operativa.

Escenario 2: Tomando como base el mismo tipo de ejercicio, se tiene como vector de entrada (V_i) producto de una interacción previa, en el que se ha aplicado una estrategia cognitivo/operativa. Después del proceso de inferencia del sistema tutor se obtiene el siguiente vector final (V_f).

Tabla 5.10 Matriz de Causalidad del Escenario 2 (9 Elementos del Tutor de Didáctica General).

	Interés	Deseo	Ayuda	Estrategia cog/op	Interrup-ción	Renuncia	Aprend.	Tiempos inactivos	Error
V_i	**1**	**1**	**1**	**1**	0	0	0	**1**	**1**
V_1	0.006693	0.006693	0.006693	1.000000	0.500000	0.500000	0.999955	0.006693	0.006693
V_2	0.992846	0.992846	0.006694	0.993943	0.000572	0.000572	0.508365	0.006694	0.006694
V_3	0.999996	0.999996	0.000000	0.526508	0.000000	0.000000	1.000000	0.000000	0.000000
V_4	0.999997	0.999997	0.000000	0.500000	0.000000	0.000000	0.999997	0.000000	0.000000
V_5	0.999996	0.999996	0.000000	0.500000	0.000000	0.000000	0.999996	0.000000	0.000000
V_f	**0.999996**	**0.999996**	0.000000	**0.500000**	0.000000	0.000000	**0.999996**	0.000000	0.000000

El vector final, indica que al incluir las estrategias, existe la posibilidad de que desaparezcan la ayuda, los tiempos inactivos y los errores. Además, se infiere algo muy importante, esta vez, el aprendizaje se presenta. Se hace notar que el interés y el deseo también continúan.

En esta sección se ejemplifican dos escenarios clasificados y detallados con los elementos de la sección anterior y sus posibles estrategias didácticas.

La matriz causal del tutor de didáctica general, apoya la evaluación del desarrollo del proceso de enseñanza-aprendizaje. Lo anterior nos permite elaborar distintas tácticas didácticas que dan vida a la intervención tutorial.

En ambos ejemplos se recomienda el uso de Estrategias Cognitivas/ Operativas. Y en ambos casos la valoración de los nueve elementos de la Tabla 5.1 se ve enriquecida con la proveniente del estilo de aprendizaje y de la motivación de estudio, así como el tipo de error que el agente manejará. Con base a esta información se determinará la estrategia didáctica dinámica y personalizada del estudiante.

Capítulo 6 "Diseño de los Objetos de Aprendizaje de ProgEst"

En este capítulo se describe el diseño de los Objetos de Aprendizaje (OA) y su implementación en el ProgEst para el dominio de programación estructurada y como ejemplo de la construcción e instrumentación de los OA aplicaremos la metodología propuesta por Muñoz-Arteaga, et al., (2006)**,** para el OA "*Estructuras de Control*".

6.1 Creación de contenidos en SCORM y la implementación en el LMS (Moodle)

Como se describió en el capítulo 3 un Objeto de Aprendizaje (OA), es la unidad mínima de aprendizaje que por sí misma tiene sentido y es independiente del contexto. Los OA cuando están formados bajo un estándar adquieren las características de durabilidad, interoperabilidad, accesibilidad y reutilización. Los contenidos se colocan en una plataforma e-Learning o Learning Management System (LMS). Los LMS pueden variar mucho y en general es posible configurar cursos, registrar alumnos y realizar diversas actividades de comunicación: foros, chats, entre otros.

6.2 SCORM (Metadatos y Paquetes)

En ProgEst se seguirán las reglas propuestas en el modelo SCORM.

Como se menciona en el capítulo 3 SCORM es un modelo de referencia que define: estándares. El modelo de empaquetamiento para las especificaciones del IMS (*Instructional Management Systems Project* /actualmente es el IMS

Global Learning Consortium quien crea y desarrolla los estándares de adopción de tecnología en e-learning) y el modelo de metadatos el LOM (*Learning Object Metadata*) de Institute of Electrical and Electronics Engineers (IEEE). Como lo muestra la Figura 6.1 el modelo SCORM está formado por 4 elementos: 1) El Overview es una visión general del todo. 2) el CAM (*Content Aggregation Model*) se ocupa del ensamblaje, etiquetamiento y empaquetamiento del contenido, 3) el RTE (*Run Time Environment*) se ocupa de las tareas relacionadas con la gestión de contenidos por parte del LMS y, por último, 4) el SN (*Sequence and Navigation*) se ocupa de la secuenciación del contenido, de los árboles de actividad, etc., (ver Figura 6.1).

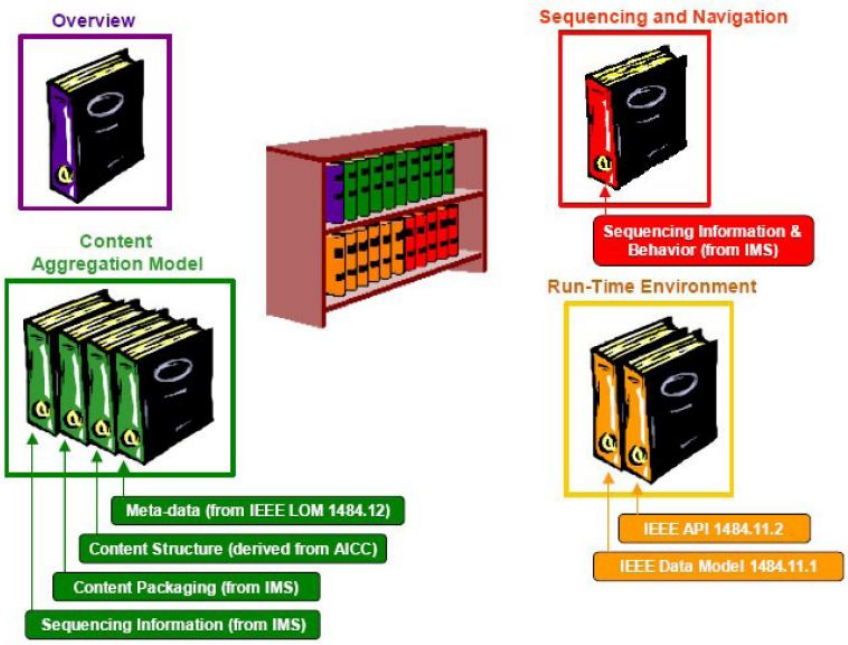


Figura 6.1 Modelo SCORM. (Landeta Etxeberría A., 2007)

Un paquete de contenido SCORM, en general, es un conjunto de archivos cuya organización está descrita en el manifiesto. En la Figura 6.2 se representa de forma conceptual un paquete de contenido y como se aprecia el manifiesto contiene toda la información relacionada con el contenido.

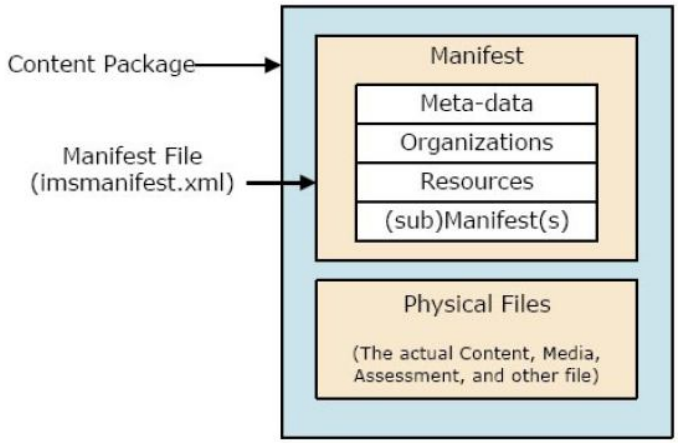


Figura 6.2 Concepto de Paquete de Contenido. (Landeta Etxeberría A., 2007)

El paquete de contenido está compuesto por los archivos, que forman un objeto y la información del objeto.

Esa *información adicional* la forman los metadatos (datos en formato público sobre el objeto), la organización (relación entre OA distintos elementos), las fuentes (objetos necesarios, archivos, para ver el contenido) y los submanifiestos (si alguno de los recursos tiene entidad propia se describe de forma anidada en un submanifiesto) si existen, y todo esto constituyen el manifiesto que se incluye en un archivo en formato XML (eXtended Mark-up Language) cuyo nombre, debe ser **imsmanifest.xml**.

Para los metadatos, LOM define 9 campos del Objeto de Aprendizaje:

- General: contiene información general como el título, la descripción, el idioma, etc.
- Ciclo de vida: se refiere a la historia y estado de evolución del paquete, a su versión, fecha, estado, etc.
- Meta-Metadata: información sobre los metadatos, por ejemplo la versión de LOM.
- Técnica: agrupa las características y requerimientos técnicos del objeto, por ejemplo su tamaño y formato.
- Uso educativo: agrupa las características pedagógicas y educativas como el nivel educativo, la dificultad, etc.
- Derechos: se refiere a los derechos de propiedad intelectual y condiciones de uso.
- Relación: se refiere a las características que relacional al objeto con otro objetos o paquete.
- Anotación: categoría que permite al autor añadir comentarios adicionales
- Clasificación: describe el objeto según uno de los sistemas de clasificación, como por ejemplo el que se emplea en las bibliotecas.

La estructura de empaquetamiento de contenido que ofrece se representa a través de un archivo o documento XML denominado IMS MANIFIEST que tiene la siguiente organización (ver Figura 6.3).

Figura 6.3 Estructura del IMS Manifiest. (Landeta Etxeberría A., 2007)

El objetivo del modelo de agregación de contenidos de SCORM es proveer un medio común de componer contenidos educativos desde diversas fuentes compartibles y reusables. Define como un contenido educativo puede ser identificado, descrito y agregado dentro de un curso o una parte de un curso y cómo puede ser compartido por diversas LMS o por diversos repositorios.

El modelo incluye especificaciones para los metadata y el Content Structure Format (CSF). Los metadata (datos sobre los datos) constituyen la clave para la reusabilidad. Describen e identifican los contenidos educativos, de manera que pueden formar la base de los repositorios. Se han especificado basándose en las recomendaciones ESE LSTC Learning Object Metadata (LOM1). Los metadata se aplican a tres niveles a los elementos de contenidos: 1) nivel ASSETS (que son la unidad más pequeña de información que es posible de representar en forma electrónica y puede ser texto, sonido, animación, página web, entre otras) este es nivel más bajo, 2) SCO (*Shareable Content Objects*), y 3) bloques de SCO y al Content Structure Format (CSF). Cabe mencionar que es necesario especificar cuáles y en qué cantidad de etiquetas se deben incluir de los metadatos para cada nivel, es por lo anterior que ADL (*Advanced Distributed Learning*) ha definido un subconjunto de etiquetas que es obligatorio (*mandatory*).

6.3 Formato para la Estructura del Contenido

El proceso de diseño y creación de un curso comprende de la construcción de un conjunto de objetos de contenidos educativos relacionados entre sí. El

objetivo del formato de la *estructura de los contenidos* es proporcionar un medio de agregación de bloques de contenidos, aplicando una estructura y asociándola a una clasificación para su presentación y un comportamiento común en cualquier LMS.

Para este proyecto de tesis se definió como caso de aplicación el curso de "*Programación Estructurada con Algoritmos en Pseudocódigo*", el dominio de conocimiento de la programación estructurada la representamos a través de la Grafo Genético (ver Figura 5.1, Capítulo 5) a partir de ésta se construyeron los siguientes OA:

- OA "*Tipos de Datos, Variables y Constantes*".
- OA "*Estructuras de Control*".
- OA "*Abstracciones*".

El modelo Content Structure Format (CSF) ha sido desarrollado a partir de las especificaciones Computer Manager Instruction (CMI) de la Aviation Industry Computed Based-Training Comitee (AICC). Posteriormente, después de una reorganización entre las especificaciones de ADL, el AICC, el IEEE y MS Global Learning Consortium, el nuevo modelo está representado en IMS Content Packaging Specification.

Un CSF es el componente que mueve un contenido educativo de un lugar a otro, pero no es suficiente por sí mismo. Es necesario agregar y guardar los contenidos en un paquete. Para ello está diseñado el content packaging. Packaging (o empaquetado), este proceso identifica todos los recursos necesarios para representar los contenidos educativos y como segundo paso es reunir esos recursos junto a un manifiesto.

Como los sugiere ADL, se aplico para el empaquetamiento el Content Aggregation Package Aplication Profile y Content Aggregation Manifest, este es una versión mejorada del CSF porque esta complementado con un nuevo método para inventariar todos los archivos requeridos para distribuir contenidos e identificar sus relaciones.

SCORM Run-Time Envoronment (entorno de ejecución)

El objetivo del entorno de ejecución de SCORM es proporcionar un medio para la interoperabilidad entre los objetos compartibles de contenidos, SCO y los sistemas de gestión de aprendizaje (LMS).

Uno de los propósitos principales de SCORM es el de construir contenidos electrónicos con propiedades para ser usados por múltiples LMS, sin importar con qué herramientas se crearon. Para que esto sea posible, debe existir un método común para desplegar un contenido, un método común para que los contenidos se comuniquen con el LMS y elementos de datos predefinidos que sean intercambiables entre el LMS y el contenido durante su ejecución.

Los tres componentes del entorno de ejecución de SCORM son:

- Launch. Es el mecanismo que define el método común para que los LMS lancen un SCO basado en web. Este mecanismo define los procedimientos y las responsabilidades para el establecimiento de la comunicación entre el contenido a mostrar y el LMS. El protocolo de comunicación está estandarizado a través del uso común del API (conjunto de funciones predefinidas de libre uso).
- LMS puede implementar la presentación de los SCO del modo que se desee, por ejemplo desarrollando un mecanismo de adaptación al usuario (mediante técnicas de aprendizaje simbólico) o bien, puede delegar esa responsabilidad al cliente permitiéndole que navegue por el curso libremente a través de menús.
- API (Application Program Interface) es el mecanismo para informar al LMS del estado del contenido y es usado para intercambiar datos entre el LMS y los SCO (por ejemplo datos de tiempo, de puntuación, etc.) La API es simplemente un conjunto de funciones predefinidas que se ponen a disposición de los SCO.

6.4 Construcción de un Objeto de Aprendizaje

Siguiendo el Grafo Genético (ver Figura 5.1, Capítulo 5) del dominio de conocimiento de "*Programación Estructurada con Algoritmos en Pseudocódigo*", desarrollaremos el diseño del objeto de aprendizaje "*Estructuras de Control*" siguiendo la metodología propuesta por Muñoz-Arteaga, et al., (2006).

Recolectar el Material del Contenido

Es necesario reunir el material necesario para crear un objeto de aprendizaje de manera que le sea más fácil tener todo listo al momento de entrar al catálogo y dejarlo disponible para quienes deseen acceder a él.

En nuestro caso el material de Estructuras de Control para el diseño se encuentra definido en la sección 4.3.3 del capítulo 4 del dominio del Sistema: Programación Estructurada con Algoritmos en Pseudocódigo.

Digitalizar el Material del Contenido

Una vez que el objeto de aprendizaje ha sido creado, es necesario traspasarlo a un formato digital para que sea posible agregarlo al repositorio.

Obtenemos los metadatos de nuestro OA

En este caso estamos hablando de datos relacionados a los OA. Los estándares para el empaquetamiento permiten definir el mecanismo para unir distintos OA creando paquetes de contenido que puedan ser compartidos entre distintos LMS. En este caso:

- Identificador: manifiestoIP
- Titulo: Introducción a la Programación
- Autor: Lourdes Sánchez
- Fecha de Publicación: Junio, 2007
- Institución: Universidad Autónoma Metropolitana
- Categoría: Informática
- Lenguaje: Español
- Éstos son algunos de los metadatos que podemos utilizar.

Con el fin de apoyar los trabajos del diseño de los objetos de aprendizaje, y siguiendo la metodología de diseño a continuación se presentan los formatos-plantillas de apoyo a la construcción del OA "*Estructuras de Control*". (Ver Tablas 6.1, 6.2, 6.3, 6.4, 6.5, 6.6 y 6.7)

Tabla 6.1 Plantilla de Análisis.

Nombre del OA	Estructuras de Control
Descripción del OA	Apoyar al estudiante a lograr el conocimiento de programación estructurada y conceptos de diseño de algoritmos en pseudocódigo utilizando las estructuras de control para de diseño de algoritmos
Nivel escolar al que va dirigido el OA	Nivel superior/ Licenciatura
Perfil del alumnos al cual va dirigido el OA (necesidad de aprendizaje)	Estudiantes de las áreas de Ingeniería.
Objetivo de aprendizaje	El alumno analizará y aplicará conceptos teóricos básicos de las estructuras de control; 1) secuencia, 2) iteración -condicional, no condicional-, 3) selección -simple, múltiple), en el diseño de algoritmos en pseudocódigo
Granularidad	Unidad

Tabla 6.2 Plantilla de Recolección.

Obtención	
Tipo de material	**Fuente**
Impresos (textos): Libros	• Luis Joyanes, Fundamentos de Programación "Algoritmos y Estructuras de Datos", Editorial Mc. Graw Hill, Segunda Edición, 1996. • Guillermo Levine, Introducción a la Programación Estructurada, Editorial Mc. Graw Hill, Segunda Edición, 1990.
Texto electrónico	http://luda.uam.mx

Tabla 6.3 Plantilla de Actividades.

Actividad			
No.	**Propósito de la actividad**	**Descripción de la actividad**	**Tipo de archivo**
1.	Que el alumno comprenda la aplicación de las estructuras de control 1) secuencia, 2) iteración -condicional, no condicional-, 3) selección -simple, múltiple, para el diseño y construcción de los algoritmos en pseudocódigo	Enseñar a resolver problemas de un modo riguroso y sistemático: metodología de análisis y diseño de algoritmos en pseudocódigo	
2.	Que el alumno comprenda la aplicación de estructuras de control (1) secuencia, 2) iteración -condicional, no condicional-, 3) selección -simple, múltiple) y sus características de cada una de ellas.		

Tabla 6.4 Plantilla de Evaluación.

Evaluación		
No.	**No. de preguntas**	**Tipo**
1.		
2.		

Tabla 6.5 Plantilla de Categoría General.

Categoría general. Agrupa la información de carácter general que identifica al OA	
Identificador	UAM3456
Título	Estructuras de Control
Entrada de catálogo	
Catálogo*	
Entrada*	
Lengua	Español
Descripción	El objetivo principal es mostrar los conceptos básicos de programación estructurada con pseudocódigo y apoyar al estudiante a lograr el conocimiento de programación estructurada y conceptos de diseño de algoritmos en pseudocódigo
Descriptor (palabra clave)	Programación Estructuras de Control
Cobertura* **	2 = Ingeniería de software 4 = Estructura de datos 5 = Lenguaje de Programación 6 = Tecnologías de Información
Estructura	Lineal: un conjunto de objetos que tienen un orden específico
Nivel de agregación	Granularidad del OA: Unidad

Tabla 6.6 Plantilla de Categoría de Ciclo de Vida.

Categoría ciclo de vida. Agrupa las características relacionadas con la historia y el estado actual del OA, así como aquellas entidades que han afectado al OA durante su evolución	
Versión	1.0
Estatus	Final
Otros colaboradores	
Rol	Autor: María de Lourdes Sánchez Guerrero
Entidad	María de Lourdes Sánchez Guerrero, Universidad Autónoma Metropolitana- Unidad Azcapotzalco
Fecha	Junio 2009

Tabla 6.7 Plantilla de Categoría Educacional.

Categoría educacional. Agrupa las características educativas y pedagógicas del OA, lo cual es importante para aprovecharlo con calidad.	
Tipo de interactividad	Activo
Tipo de recurso de aprendizaje	Texto, Ejercicios, Cuestionarios y Ejemplos.
Nivel de interactividad	Muy alto
Densidad semántica	Media
Fines de uso del usuario	Estudiante
Contexto	Universidad
Edad	19 años
Dificultad	Medio
Tiempo promedio de aprendizaje	20 minutos por ejercicio a lo largo de un trimestre
Descripción	Seguir las instrucciones
Lengua	Español

Para continuar con este proceso es necesario integrar cada uno de los componentes del OA (identificados en la fase II de la metodología propuesta, ver Anexo 1.

Construimos el archivo XML (imsmanifest.xml)

Además de que constituido de tres partes:

- Metadatos (Metadata).
- Organizaciones (Organizations).
- Recursos (Resources).

Archivo **imsmanifest.xml**

```
<?xml version="1.0" encoding="UTF-8" ?>
 <manifest >
```

```
        <metadata>
              <general>
               <identificador>manifiestoIP</identificador>
               <titulo>
                   <langstring>Estructuras de Control</langstring>
               </titulo>
               < author>
                   <autor>Lourdes Sánchez</autor>
               </author>
              <categoria>
                   <catalog>Informática</catalog>
                    <entry>
                     <Institucion>Universidad Autónoma
Metropolitana</Institucion>
                    </entry>
              </categoria>
              <language>es</:language>

          </metadata>

<organizations>
<organization identifier=" UAM-AZC060707">
        <title>Introducción a la Programación</title>
        <item identifier="Tema1_0">
              <title>Capitulo 1</title>
              <item identifier="Topic11" resourceref="SubTopic11">
              <title>Algoritmos</title>
        </item>
</organizations>

<resources>
<resource identifier="SubTopic11" type="webcontent"
href="topics/algoritmo.html">
        <file href="topics/algoritmo.html" />
 </resource>

</ manifest >
```

6.6 Los LMS (Learning Management System)

Los LMS son sistemas con capacidad de integrar los servicios para publicar información y herramientas web que soporten las experiencias de aprendizaje. La información que presentan puede ser de un nivel muy bajo de agregación o totalmente desagregada, como una imagen, en un nivel mayor de agregación publicar un OA, o en el más alto nivel de agregación equivalente a un curso.

Actualmente, los LMS no revisan el nivel de agregación que tiene la información que se publicará, interpretan el resumen de la descripción genérica y codificada de la información y se almacena en el lenguaje XML, este resumen es conocido como metadata (datos de los datos) y nos permite que el LMS “reconozca” al OA para así cumplir algunos objetivos del XML como la comunicación entre aplicaciones (LMS) y descripción de paquetes y sus interdependencias OA.

Si hace una analogía de los metadata, éstos serían la etiqueta de los productos alimenticios enlatados donde se refieren contenido, caducidad, medida, etcétera.

Sistemas de Gestión de Aprendizaje Virtuales

Los sistemas de gestión educativa utilizan fundamentos teóricos de distintas disciplinas como comunicación, pedagogía, filosofía, psicología, ciencias computacionales y principalmente fundamentos de las principales corrientes teóricas de aprendizaje con el fin de mejorar el proceso de enseñanza aprendizaje.

La tecnología educativa, por sus continuos adelantos y desde diferentes perspectivas, se mantiene como un pilar dentro de las configuraciones de educación a distancia virtual, o e-learning, tanto investigadores como profesionistas se apoyan en los últimos productos de la tecnología educativa.

El antecedente de los sistemas de gestión de aprendizaje era la publicación de contenidos didácticos en Internet mediante páginas web, pero no era suficiente cuando se necesitaba utilizar funciones más complejas como el registro de los avances de los estudiantes, el control de usuarios y perfiles, videoconferencias, correo electrónico, portafolios, etcétera.

La integración de todas las funciones mencionadas fue un software conocido de manera genérica como Content Management System (CMS), y su fin principal es servir de contenedor de cursos.

Los CMS con fines educativos son los LMS (*Learning Management System*), un LMS es una aplicación de software local o software basado en tecnologías web usada para planear, implementar y asegurar un proceso de enseñanza-aprendizaje. Normalmente un LMS provee un instructor y un proceso para entrega de contenidos, monitoreo de la participación de los estudiantes y una evaluación.

Actualmente existen los LCMS (*Learning Content Managementn System*), estos sistemas de gestión de contenidos de aprendizaje son plataformas mayores que

permiten personalizar los recursos para cada alumno estructurando los ambientes facilitan los procesos y prácticas educativas con el apoyo de cursos, materiales y contenidos en línea.

La arquitectura de los LCMS está basada en objetos de contenido (también llamados Objetos de Aprendizaje), el objeto es reutilizable en cursos, currícula y organizaciones, los contenidos se pueden publicar en diferentes formatos y no están asociados a algún tipo de navegación particular, estas tareas las realiza el LCMS mediante etiquetas de XML y siguiendo ciertos estándares como SCORM.

6.7 LMS (Moodle)

El LCMS en el cual se basa este proyecto es MOODLE (*Modular Object-Oriented Dynamic Learning Environment*). Por ser un LMS de código libre. Es un CMS originalmente desarrollado por Martín Dougiamas en 2002, y usado por cientos de instituciones educativas en todo el mundo, una de sus fortalezas es la amigabilidad con el administrador, lo que facilita su aprendizaje e implantación, además la cantidad de opciones hace que la personalización de un curso específico sea más sencillo.

Una opción es contratar una compañía de administración de sitios web (hosting: es el servicio que provee a los usuarios de Internet un sistema para poder almacenar información, imágenes, vídeo, o cualquier contenido accesible vía Web.) para que le administre su CMS y pueda centrarse en otras actividades, también existe la opción de instarlo y administrarlo uno mismo, si se decide por la segunda opción a continuación se describe el proceso de instalación:

Como requisito de instalación MOODLE es necesario:

1. Un servidor web, con Apache (webserver), o el HS de windows.
2. Una instalación de PHP (Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools)).
3. Una base de datos preferentemente MYSQL (es un Sistema Gestor de Bases de Datos (SGBD), que nos permite de forma fácil y sencilla almacenar y recuperar información de forma relacional).

Si no tiene aún instalado Apache (el servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP), PHP y MYSQL, se recomienda utilizar EasyPHP este software incluye los tres componentes.

Cabe mencionar que es necesario que en la computadora destino **no** esté activa ninguna versión de Apache. PHP o MYSQL.

Para obtener EasyPHP

Es posible obtener una copia de PHP en;

http<.//prdownloads.sourceforgel.neet/quickeasyphp/easyphp1-7<-setup. Exe´.download.

Si la instalación easyphp fue exitosa, aparece la aplicación en la barra de tareas, y se ahí también se puede obtener una copia de PHP.

Como crear la Base de Datos

El siguiente paso es asociar una base de datos para el uso de Moodle: dé clic derecho en el icono E, seleccione DB Management y después seleccione pm.-y Admin, la cual proveerá una interfaz con la que podrá crear una base de datos y permisos de usuario, además de asignar la cuenta de administrador-

Solamente teclee "*moodle*" en el campo y presionará el botón Create, así tendrá instalado el servidor php.

Como Instalar Moodle

Si está listo para instalar Moodle, baje la versión más reciente en el sitio http//download.moodle.org/, y desempaque el archivo.

La carpeta donde se encuentran los archivos de Moodle desempacados se copiarán a la carpeta C:/Archivos de programa/ EasyPHP 1-8/www.

Haga una carpeta con el nombre de "moodledata". Active a la carpeta "moodle", copie el archivo config.dist.php y renómbrelo config. Php.

Para editar el Archivo config. php

Use un editor de texto como Block de notas y procure no dejar espacios en blanco al final.

El archivo debe quedar asi:

$CFG>dbtype= ´mysql´;

$CFG>dbhost= ´localhost´;

$CFG>dbname= ´moodle´;

$CFG>dbuser= ´root´;

$CFG>dbpass= ´´;

$CFG>dbpersist= true;

$CFG>prefix= ´mdl_´;

$CFG>wwwroot= ´http//127.0.0.1/moodle´;

$CFG>dirroot= ´C: Program Files EasyPHP www moodle´;

$CFG>dataroot= ´C moodledaa;

No es necesario modificar el resto de las líneas. De esta forma ya está listo para usar moodle. Los pasos básicos son ejecutar EasyPHP y seleccionar “moodle”, en la pantalla de inicio de Moodle está el ícono de creación de cursos.

Caso de Estudio

Es importante mencionar que los objetos de aprendizaje solo se pueden aprovechar sus características de durabilidad, interoperabilidad, accesibilidad y reutilización, a través de una plataforma de aprendizaje con soporte tecnológico LCMS

El LCMS podrá publicar el recurso educativo si éste contiene su respectivo manifest.xml (archivo metadato). El proceso de publicación es el siguiente:

1. Utilizar un LCMS que use el estándar metadata SCORM.
2. Crear un curso en el LCMS.
3. Añadir el recurso educativo, es decir, el OA, mediante la opción “Agregar recurso”.
4. Navegar en el OA y realizar las actividades de aprendizaje.

Como se aplica un LCMS aplicando el estándar metadata SCORM.

Para este proyecto aprovecharemos las ventajas que nos proporciona el LCMS MOODLE por ser software libre, los pasos para ejecutarlo son: botón de inicio, todos los programas, EasyPHP y enseguida aparecerá un icono E en la barra de herramientas con lo cual el software estará activado.

Como paso siguiente es abrir una página de explorador de Internet (FireFox o Internet Explorer) y pulsar http://localhost/.

En la pantalla del explorador aparecerán 25 opciones relacionadas con el curso que se está creando, como mostrar calificaciones, mostrar informes de actividad, etcétera. Después presione el botón de “guardar cambios”, aparecerá un área de la plataforma nombrada “curvar” en la que son visibles todas las semanas del cuso (según las fechas capturadas).

Una vez llenados los datos de nombre y resumen se presionará el botón “elegir o actualiza un paquete SCORM”, en esta etapa el proceso es el siguiente:

1. Se pulsa el botón “subir el archivo”.
2. Cuando sea visible el archivo seleccionado se podrá realmente seleccionar con la opción “elegir” del lado derecho y finalmente deberá pulsarse el botón “guardar cambios”.

Con el procedimiento anterior es posible tener disponible un recurso educativo en la forma de OA.

Para el empaquetamiento del OA (RELOAD)

RELOAD (*Reusable e-Learning Object Authoring Delivery*) es una herramienta para facilitar el uso de especificaciones para la interoperabilidad de contenidos por las tecnologías de aprendizaje provistas por ADL (SCORM). La herramienta nos permitirá hacer que los contenidos educativos electrónicos comunes se conviertan en contenidos reutilizables (portables) por cualquier LMS compatibles con SCORM, como Moodle, Calroline, Dokeos. ATutor, Etcétera, esta interoperabilidad se logra mediante la adición de información relacionada (metadata) por medio de RELOAD.

La herramienta RELOAD consta de edición de metadata, empacado de contenidos. Para adquirir la herramienta RELOAD sólo es necesario acceder al sitio http//www.reload.ac.uk, donde aparece la liga http://www.cetis.ac.uk/members/telcert/downloads/Reload_Dist.zip, de donde se puede obtener una copia de esta herramienta. Para instalar, basta con ejecutar el archivo recibido.

Pasos para el empaquetado de contenidos empleando RELOAD

Los pasos generales a seguir por los empacadores de contenidos son:

1. Conjuntar (en una carpeta con el nombre apropiado) todos los recursos (imágenes, animaciones, hojas de estilo, hojas de formato, páginas web, etcétera).
2. Aplicar la herramienta RELOAD.
3. Importarlo a un LMS o a un repositorio digital.

Producir un paquete de contenidos.

Iniciar el editor RELOAD

File > New> ADL SCORM 1.1 Package> Seleccionar carpeta con los recursos.

Para Crear la Organización

La organización es el árbol del curso. En caso de tener más de una página, esa organización contendría la estructura en la que se presentará. Para crearla sólo se necesita activar el botón derecho sobre Organizations y seleccionar Add Organizations.

Una vez activada esa Organización se "vinculan" con los archivos que serán explorados por el usuario. En este ejemplo son cuatro archivos html. La página que enlazaremos (o vincularemos) será la 1_P1-html, y así sucesivamente.

Expandimos la sección de Resources y seleccionamos 1_P1.html. Esto activara una pequeña tabla en la parte inferior de la ventana. Dicha tabla contiene una fila donde aparece SCORM Type/asset. Sólo se tiene que seleccionar la opción de SCO en lugar de asset por cada página.

Capítulo 7 “Ejemplos de uso del ProgEst”

En este capítulo desarrollamos un ejemplo de uso del sistema ProgEst tomando como caso de estudio, la evaluación del escenario “*Estructuras de Control*”, con el fin de ejemplificar paso a paso la ejecución del ProgEst.

Ejemplo

En este ejemplo el alumno se registra por primera vez (ver Figura 7.1) en el sistema, al concluir el sistema le asigna un número de usuario y password, a continuación el Sistema lo regresa a la pantalla de inicio con el fin de registrar los datos (usuario y password). Como paso siguiente se activa el cuestionario de Estilos de Aprendizaje (ver Figura 7.2), al finalizar el llenado del cuestionario, el sistema evalúa las respuestas y determina el estilo de aprendizaje del alumno (ver Figura 7.3), para este ejemplo el resultado es que el alumno es de tipo ***“teórico”***.

Figura 7.1 Pantalla de Registro de Datos.

A continuación se activa el cuestionario de Motivación de Estudio (ver Figura 7.4), al finalizar el llenado del cuestionario el sistema evalúa las respuestas y determina el tipo de motivación de estudio del alumno, para este ejemplo el alumno tiene motivación externa. (Ver Figura 7.5)

Los datos de los cuestionarios y el registro son guardados en la Base de Datos para que más tarde se consulten por el sistema.

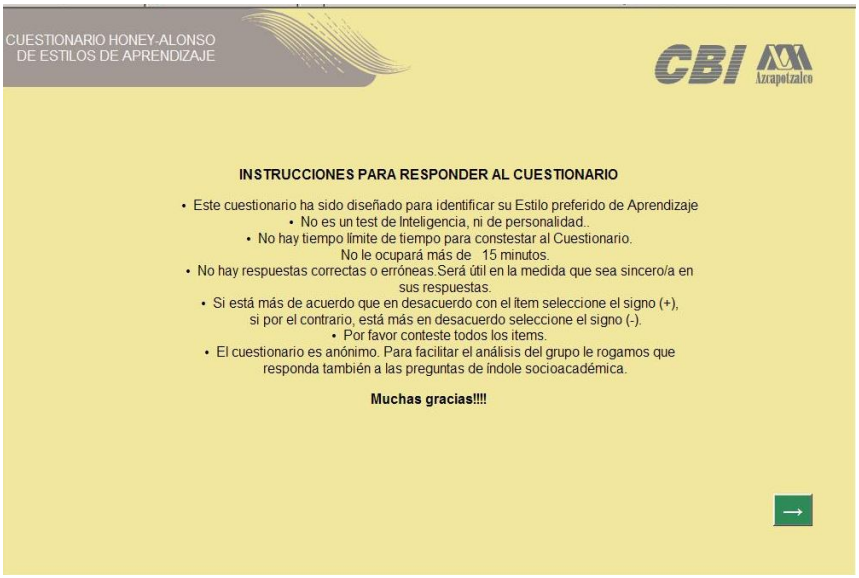


Figura 7.2 Pantalla Cuestionario Estilos de Aprendizaje.

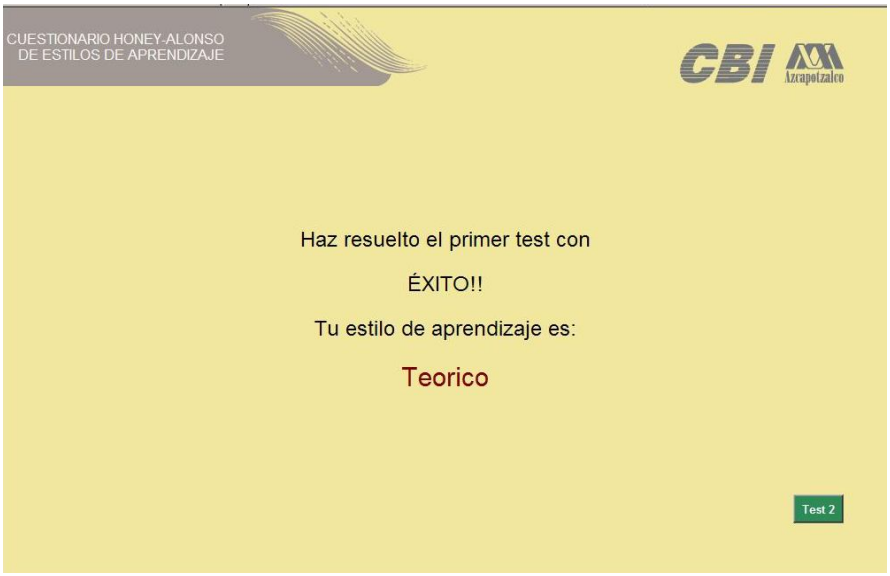


Figura 7.3 Pantalla de Resultado del Cuestionario de Estilo de Aprendizaje.

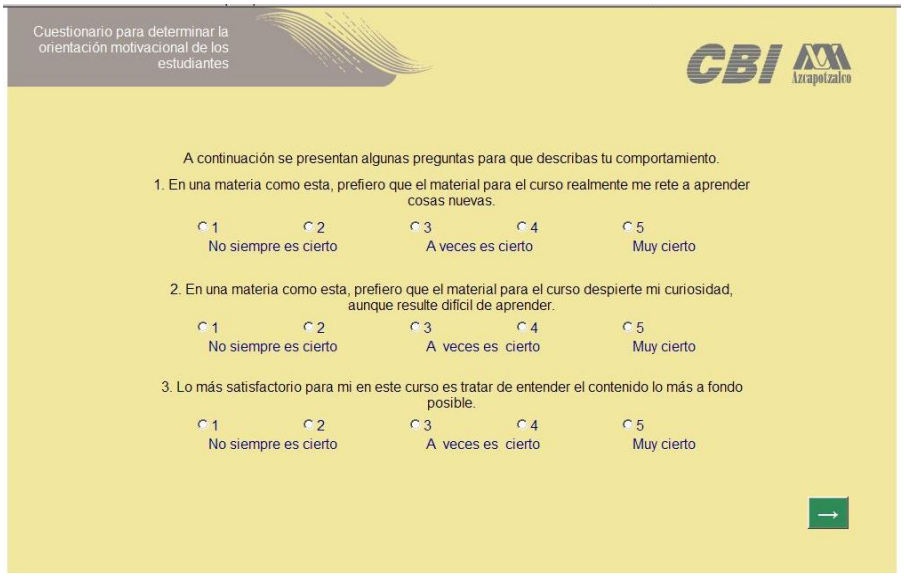


Figura 7.4 Pantalla de Cuestionario de Motivación de Orientación de Estudio.

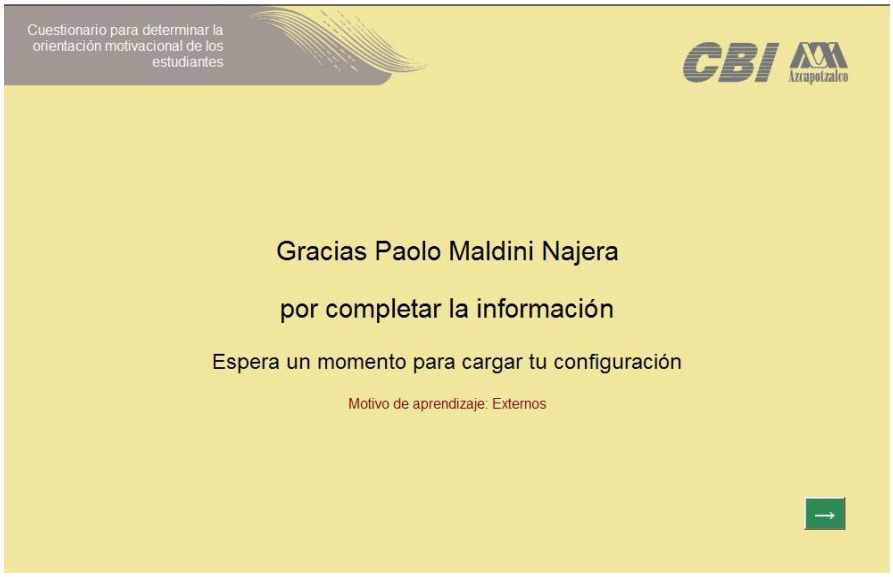


Figura 7.5 Resultado de Cuestionario Motivación de Orientación de Estudio.

Para continuar con el proceso, se activa el escenario "*Estructuras de Control*" para este ejemplo. (Ver Figuras 7.6 y 7.7)

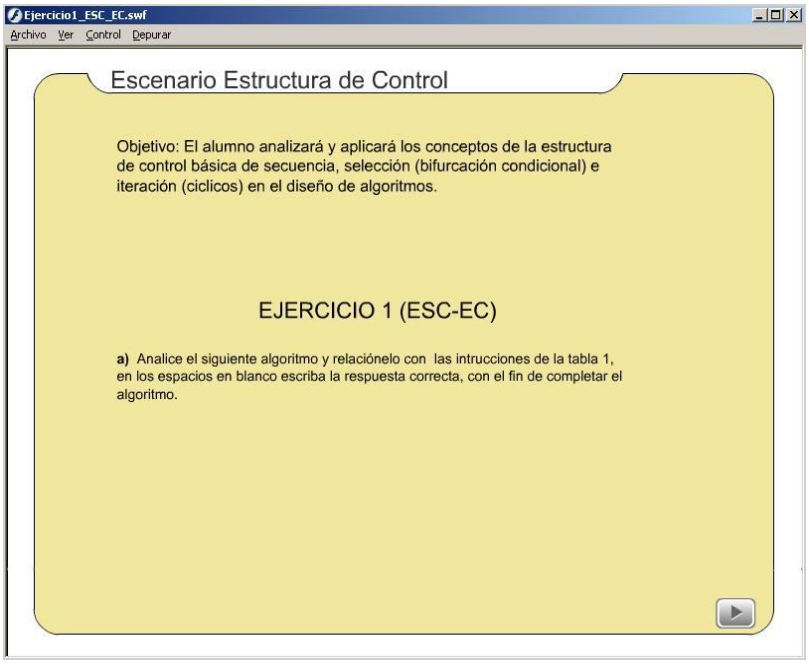


Figura 7.6 Pantalla Principal del Escenario de Estructuras de Control.

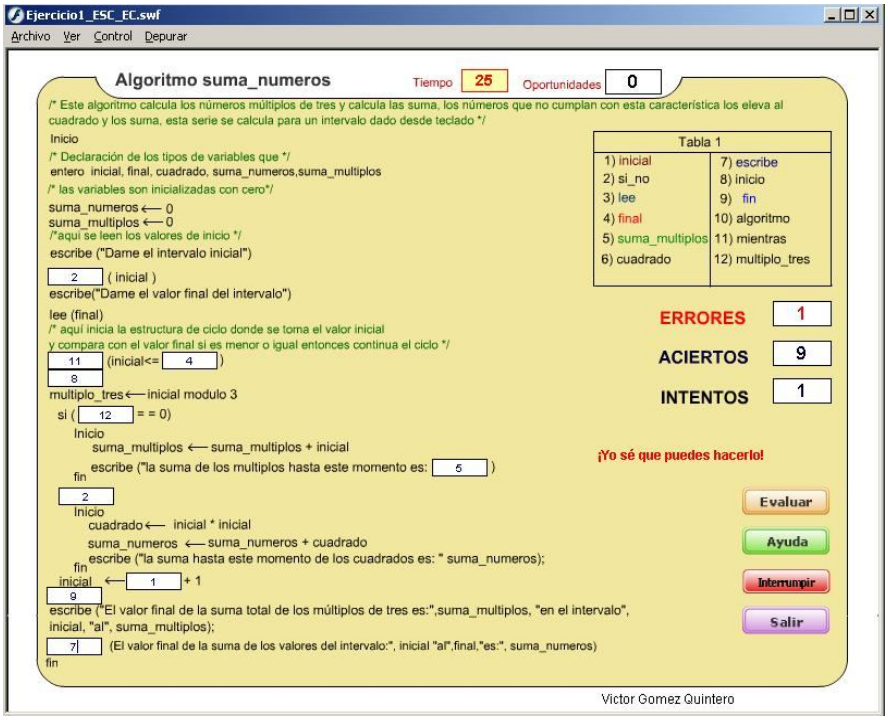


Figura 7.7 Pantalla de Presentación del Escenario de Estructuras de Control.

Al resolver el escenario se pueden presentar varios casos, de los cuales se describirán a continuación:

a) Si el alumno solicita ayuda, de acuerdo a las tácticas didácticas explicadas en el capítulo 5 (Sección 5.7) y al tipo de motivación de estudio del alumno (ver Figura 7.8), el sistema a través de la interfaz despliega el objeto de aprendizaje pertinente del escenario, en este caso, Estructuras de Control para el estilo teórico (ver Figura 7.9), permitiendo a alumno regresar al escenario al término de la consulta.

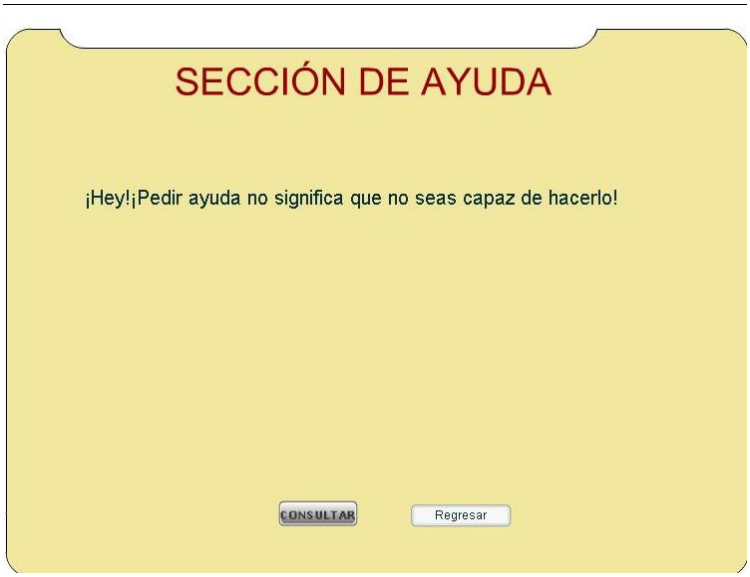


Figura 7.8 Pantalla de Ayuda del Escenario de Estructuras de Control para el Estilo Teórico y con Motivación Externa.

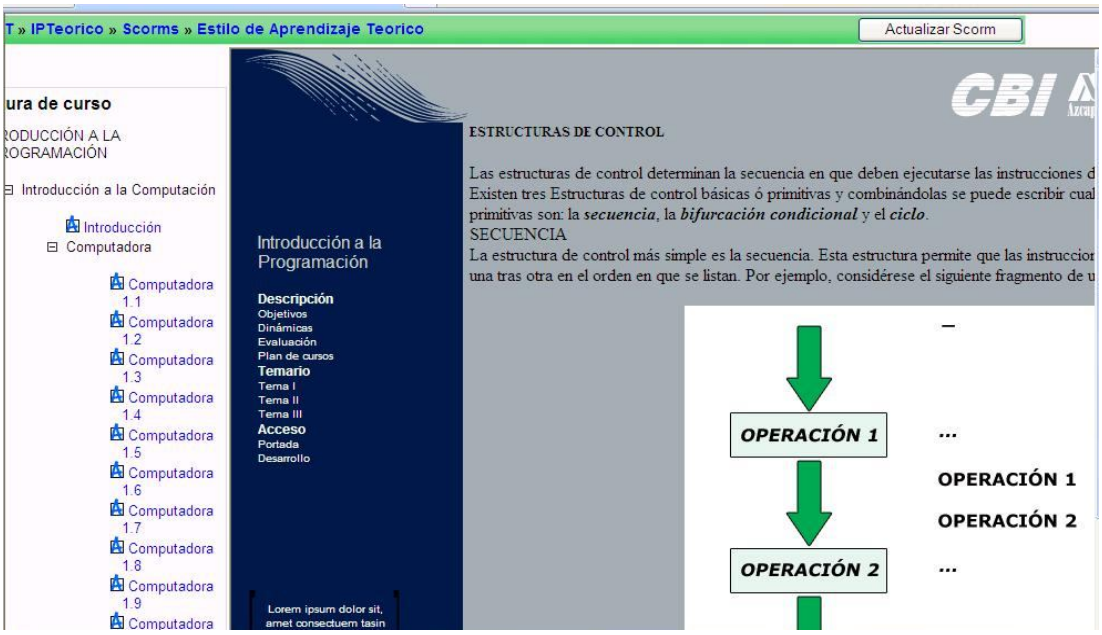


Figura 7.9 Pantalla de Presentación del Objeto de Aprendizaje del tema de Estructuras de Control para el Estilo Teórico.

b) Si el alumno comete algún tipo de error de acuerdo a la clasificación mencionada en el capítulo 5 (Sección 5.7), las tácticas didácticas son distintas para cada tipo de error:

- Error Leve.- En este caso el Sistema permite al alumno continuar resolviendo el escenario, siempre y cuando las 3 oportunidades para resolver el escenario no se hayan agotado. (Ver Figura 7.10)

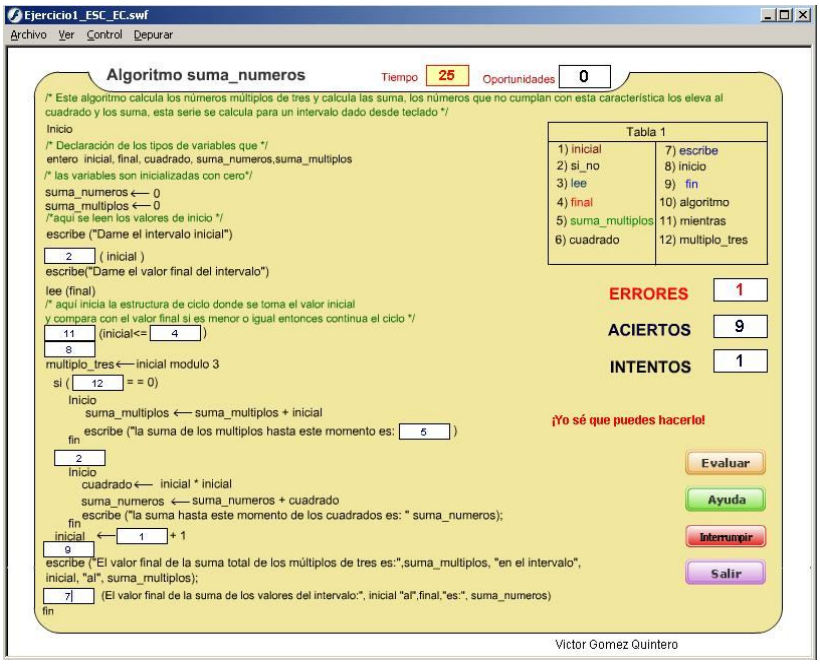


Figura 7.10 Pantalla de Resultado del Escenario al presentarse un error leve.

- Error Grave.- En este caso el sistema decide mostrar a través de la interfaz el material del tema de Estructuras de Control para consulta del alumno. Al término de la consulta, el sistema permite al alumno iniciar de nuevo el ejercicio. (Ver Figura 7.11)

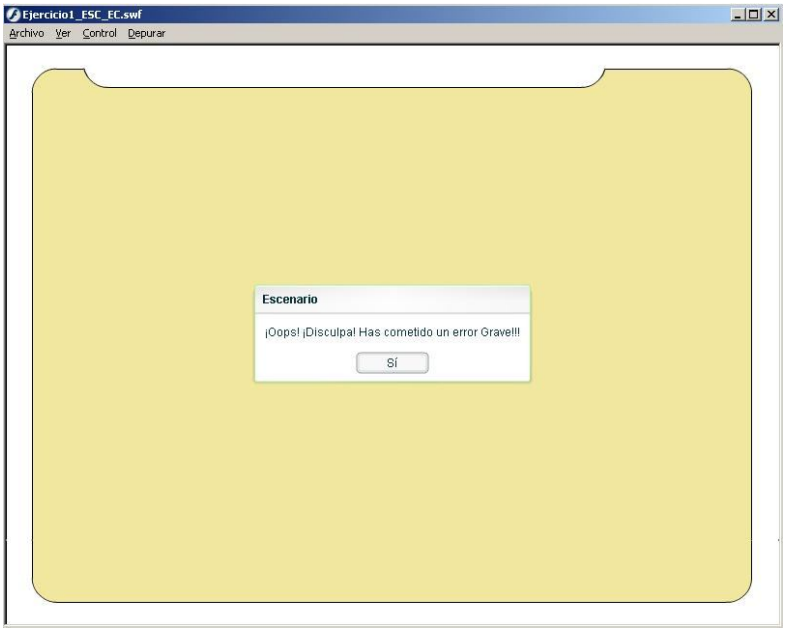


Figura 7.11 Pantalla de Resultado del Escenario al presentarse un error grave.

- Error Fatal.- En este caso el sistema decide dar por terminado el escenario y a continuación muestra el material del dominio de Programación Estructurada con Algoritmos en Pseudocódigo de acuerdo al estilo de aprendizaje determinado previamente, para que el alumno repase el material. (Ver Figuras 7.12 y 7.13)

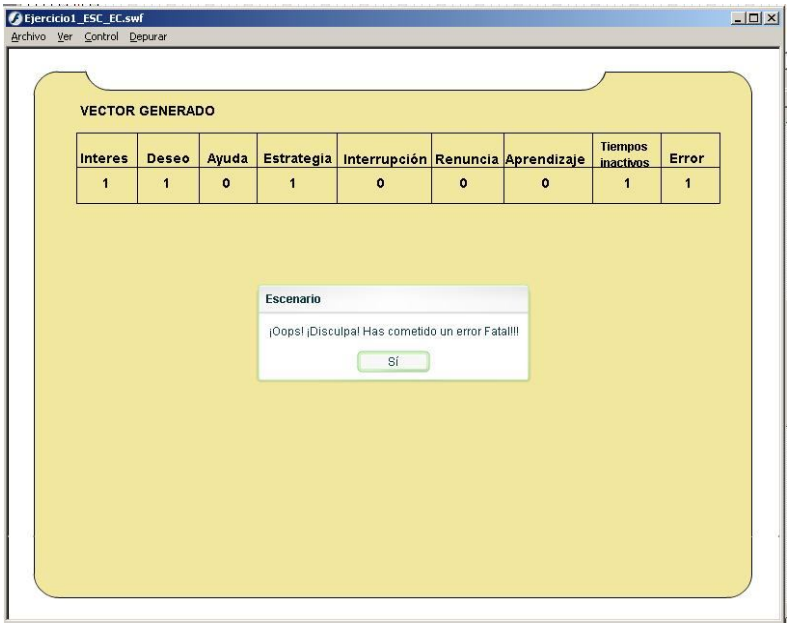


Figura 7.12 Pantalla de Resultado del Escenario al presentarse un error fatal.

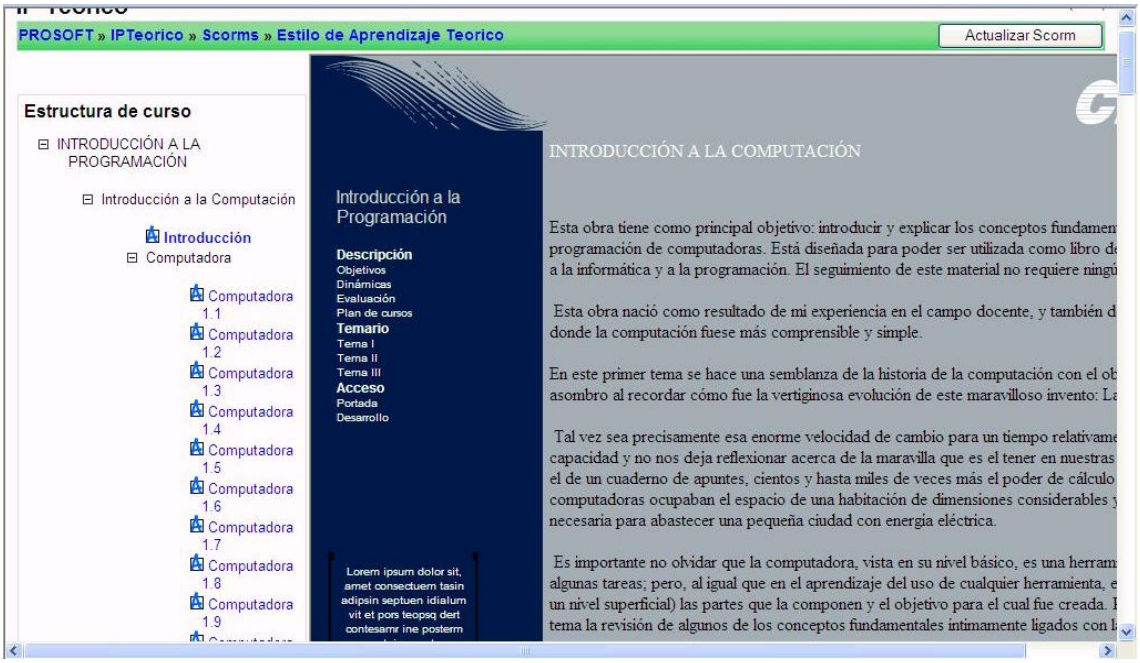


Figura 7.13 Pantalla de Presentación del Objeto de Aprendizaje del tema del dominio de Programación Estructurada con Algoritmos en Pseudocódigo para el Estilo Teórico.

c) Si el alumno solicita una interrupción, se suspende temporalmente el reloj que controla las oportunidades para resolver el escenario permitiendo al alumno realizar otra actividad (consultar el material del tema, entre otras) para que después pueda regresar al escenario y continuar. (Ver Figura 7.14)

Figura 7.14 Pantalla de Resultado de Interrupción del Escenario de Estructuras de Control.

d) En este escenario el Sistema permite al alumno contar con tiempos inactivos durante la ejecución del escenario, en este caso, en el segundo tiempo inactivo de 30 segundos, el sistema despliega a través de la interfaz el letrero de “¡Hey!, Es tiempo de comenzar a trabajar”, esto de acuerdo a las tácticas didácticas determinadas en el capítulo 5 (Sección 5.7). A partir del tercer tiempo inactivo se resta una oportunidad de las 3 oportunidades que tiene el alumno de resolver el ejemplo.

Capítulo 8 "Desarrollo del Sistema Inteligente de Aprendizaje con Objetos de Aprendizaje (ProgEst)"

En este capítulo se explica el proceso de desarrollo del **ProgEst** de acuerdo a la arquitectura propia diseñada para este trabajo de tesis. Antes de iniciar con la descripción de la arquitectura del sistema se aborda el diseño y características de la interfaz del Sistema.

8.1 Diseño de la Interfaz del ProgEst

El diseño de la ***Interfaz del ProgEst*** es realizado con las características de la retícula de contenido, de uso y aplicación de la "*Pantalla de Presentación del Sistema*" tal como se muestra en la Figura 8.1. Para mayor información del diseño y aplicación de la ***Intefaz del ProgEst***. (Ver Anexo 2)

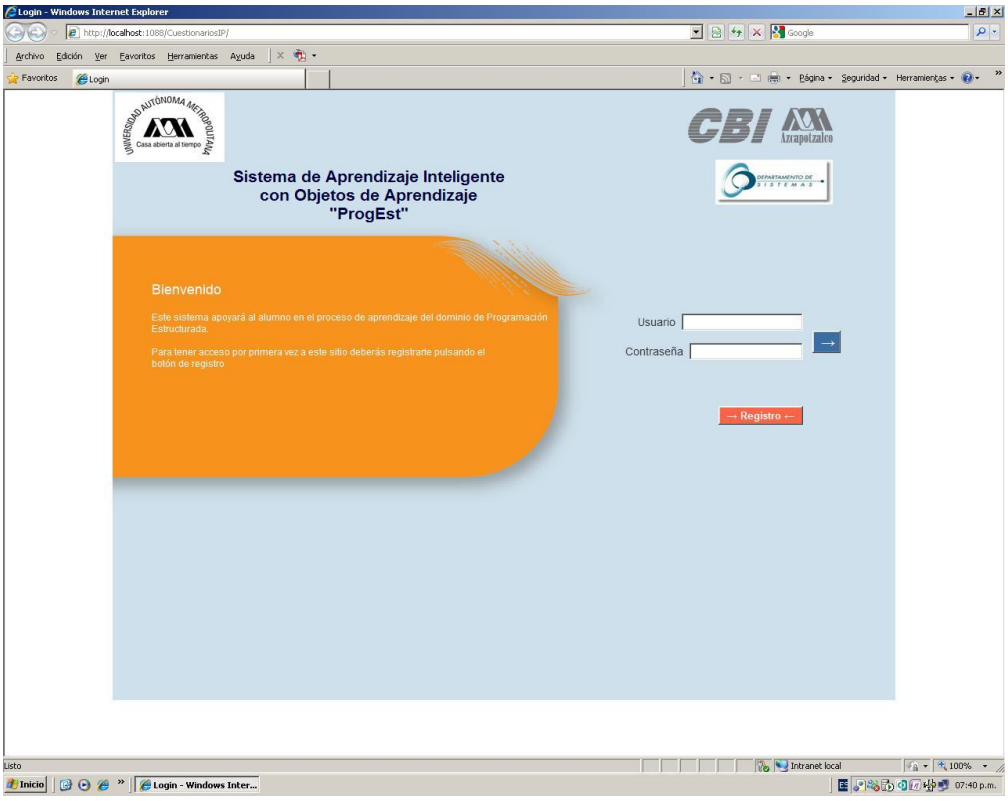


Figura 8.1 Pantalla de Presentación del ProgEst.

La aplicación de la "*Pantalla de Presentación*" se usará en los cuestionarios CHAEA y Motivación de Estudio. Así mismo en la presentación del material de Programación Estructurada con Algoritmos en Pseudocódigo, los colores dependen del estilo de aprendizaje (Activo, Reflexivo, Teórico y Pragmático) para cada usuario con respecto a la psicología del color. (Velasco-Santos, Laureano-Cruces, Mora-Torres y Sánchez-Guerrero, 2009)

8.2 Arquitectura del ProgEst

Dentro de la descripción de los componentes, cabe mencionar que para el desarrollo e instrumentación de la aplicación del **ProgEst** se contó con la infraestructura de cómputo descrita a continuación en la Tabla 8.1.

Tabla 8.1 Infraestructura de cómputo utilizada.

	Hardware	Sistema Operativo
Características del Servidor Linux (donde se encuentra implementado la plataforma LMS (Moodle) para contener los objetos de aprendizaje).	• DELL Optiplex 745 • CPU. • 2 Prcesadores: Intel® Pentium® D a 3.40 GHz • Memoria RAM: 1 GB • Disco Duro: 80GB. • 2 Tarjetas de Red: Broadcom 5754 Gigabit Ethernet LAN Solution 10/100/10006	• Fedora Linux 7. • Apache 2.0. • PHP versión 5.2.6. • Mysql 5.0.45. • Moodle 1.5.2.
Características del Servidor Microsoft (donde se encuentran implementados los cuestionarios CHAEA y Orientación).	• DELL PowerEdge 700 • CPU • 2 Procesadores: Intel® Pentium® 4 a 3.40 Ghz • Memoria RAM: 1GB • Disco Duro: 160GB • Tarjeta de Red: Intel® PRO/1000 CT Network Connection	• Windows 2003 Server R2 con Service Pack 2 • Servidor de IIS 6.0 • SQL Server 2000

Con el fin de facilitar el análisis y diseño del sistema, el **ProgEst** se subdividió en tres componentes:

- Registro del Alumno y Cuestionario de Estilos de Aprendizaje.
- Cuestionario de Motivación de Estudio (determina la orientación de motivación de estudio).
- Sistema de Aprendizaje Inteligente con Objetos de Aprendizaje.

La Figura 8.2 muestra el diagrama que representa la Arquitectura General del Sistema **ProgEst.**

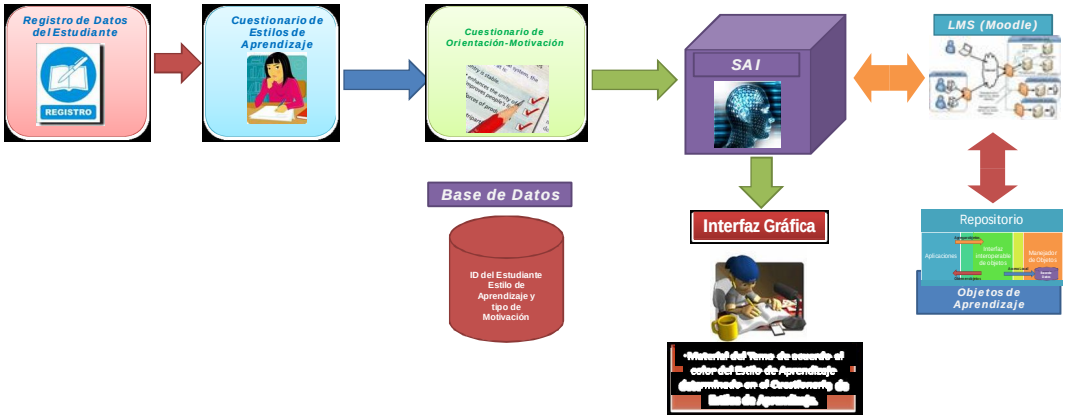


Figura 8.2 Arquitectura General del Sistema ProgEst.

8.2.1 Componente 1: "Registro del Alumno y Cuestionario de Estilos de Aprendizaje"

Este Componente tiene como objetivo registrar al alumno en el sistema y aplicar en línea el cuestionario para identificar el estilo de aprendizaje del estudiante.

Como primer paso del ProgEst está el Registro de los Datos del Alumno (ver Figura 8.1), después de esto se activa el cuestionario de Estilos de Aprendizaje. Éste se aplicará al estudiante que por primera vez se registra en el Sistema, después de finalizar el registro de los datos, se activará la pantalla del cuestionario de Estilos de Aprendizaje (Ver Figura 8.4)

Para identificar el Estilo de Aprendizaje del estudiante nos apoyamos en el Cuestionario Honey-Alonso de Estilos de Aprendizaje (CHAEA) de Alonso y Honey, (1994).

El Componente 1 se fundamenta en la arquitectura propuesta para nuestro sistema y está representada por la Figura 8.3. Éste componente se programó en lenguaje C# y con **ASP.NET** (Active Server Pages (**ASP.NET)** que es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web, servicios web XML) y se emplea SQL Server para el diseño de la Base de Datos del Sistema.

Este módulo del Sistema está bajo la plataforma de Windows 2003 Server R2 con Service Pack 2, Servidor de IIS 6.0 y SQL Server 2000. (Ver Figura 8.4)

Figura 8.3 Arquitectura del Componente 1.

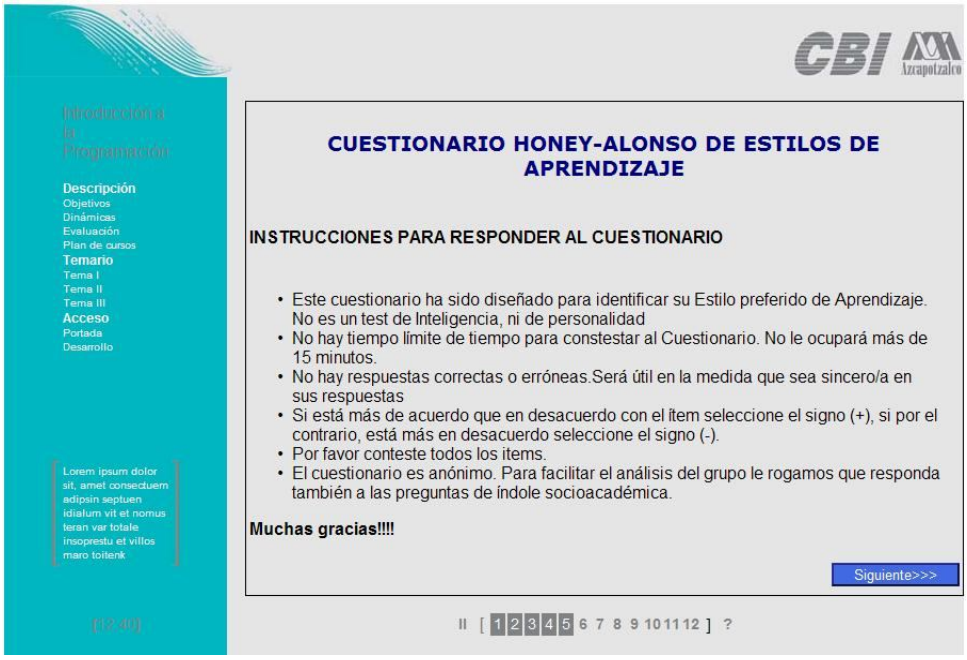


Figura 8.4 Pantalla de presentación del Cuestionario CHAEA.

El sitio Web está formado de 13 aplicaciones (páginas) en ASP.NET, donde la aplicación *página principal (master.page*) es quien administra la comunicación con las demás aplicaciones del programa. Asimismo para guardar los datos de los resultados de la aplicación se instrumentó la Base de Datos en SQL Server, quien es controlada por la función (*Chaea.cs*) del programa.

En la Figura 8.5, se muestra el diagrama de comunicación entre las páginas web en aspx que forman el cuestionario y la base de datos.

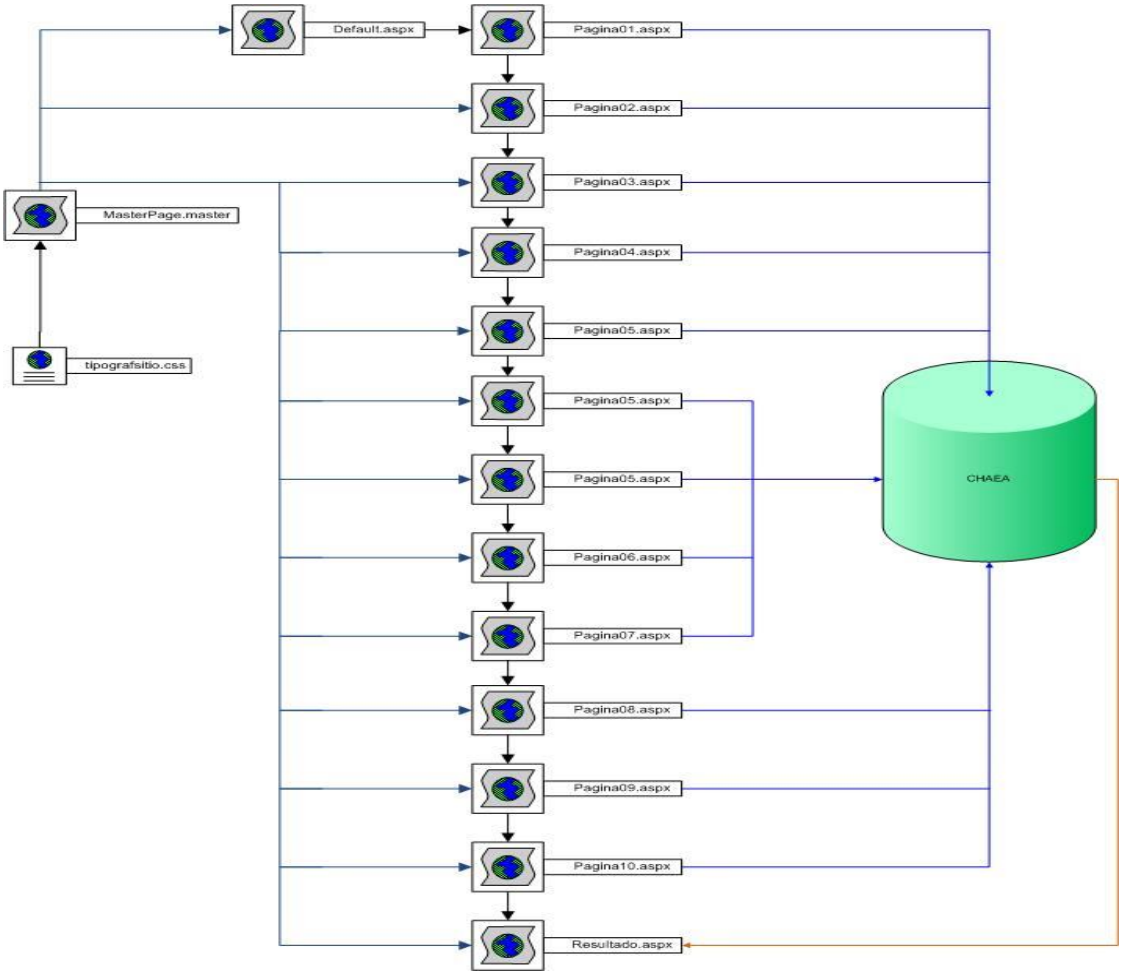


Figura 8.5 Diagrama de comunicación entre las páginas del Cuestionario CHAEA.

Para mayor explicación de este Componente se puede consultar el Anexo 3, donde se describe el "*Registro del Alumno y Cuestionario de Estilos de Aprendizaje*" y los programas (código) para la implementación de este componente.

8.2.2 Componente 2: "Cuestionario de Motivación de Estudio"

En este componente se implementó el Cuestionario de Motivación de Estudio como un instrumento para determinar la Motivación de Orientación de Estudio en el alumno, (Pintrich, Smith, García y Mckeachie, 1991), a través de los objetivos internos externos del mismo. (Ver Figura 8.7)

En **ProgEst** nos apoyamos en este instrumento para determinar el objetivo interno y externo con el fin de articularlo con el modelo de didáctica general, (Laureano-Cruces, et al., 2004a) y (Laureano-Cruces, et al., 2008b).

La Arquitectura propuesta de este componente se muestra en la Figura 8.6.

Figura 8.6 Arquitectura del Componente 2.

Este Componente se programó en lenguaje C# y con **ASP.NET** como *framework* para la aplicación de este Sistema y para esta aplicación la Base de Datos que utilizamos es SQL Server 2000. (Ver Figura 8.7)

Este módulo del Sistema está bajo la plataforma de Windows 2003 Server R2 con Service Pack 2, Servidor de IIS 6.0 y SQL Server 2000.

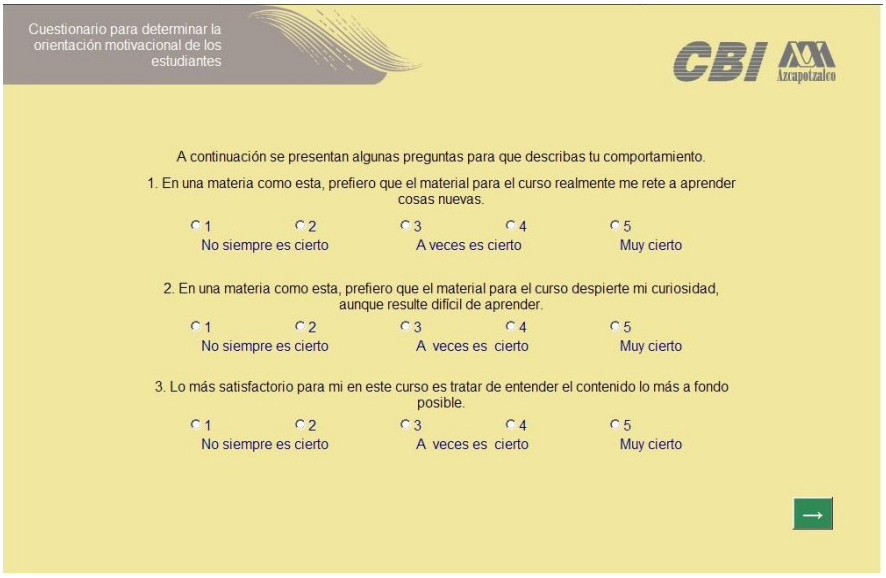


Figura 8.7 Pantalla de Presentación del Cuestionario de Motivación de Orientación

Este Cuestionario de Motivación de Estudio se activa, como paso siguiente en el Sistema, en el momento que el estudiante finaliza el cuestionario CHAEA. Sus datos de salida, al igual que el cuestionario CHAEA, son guardados en la base de datos para ser consultados por el módulo del Sistema de Aprendizaje Inteligente (SAI).

Para mayor información del código de los programas consultar el Anexo 3.

8.2.3 Componente 3: Sistema de Aprendizaje Inteligente con Objetos de Aprendizaje

El Componente 3 es el módulo central del sistema **ProgEst**, debido a que dentro de éste se trabaja de manera articulada el módulo experto, el módulo tutor y el modelo del estudiante. Lo anterior para aplicar las estrategias adecuadas de aprendizaje del dominio "*Programación Estructurada con Algoritmos en Pseudocódigo*". El dominio se encuentra contenido en objetos de aprendizaje presentados a través de la interfaz del Sistema.

El Componente 3, es la parte medular del **ProgEst**, porque aquí se evalúa y se toman las decisiones del Sistema. Para construir este módulo se propuso la arquitectura que se muestra en la Figura 8.8.Esta arquitectura está compuesta de varios submódulos que proveen los datos al SAI para la toma de decisiones.

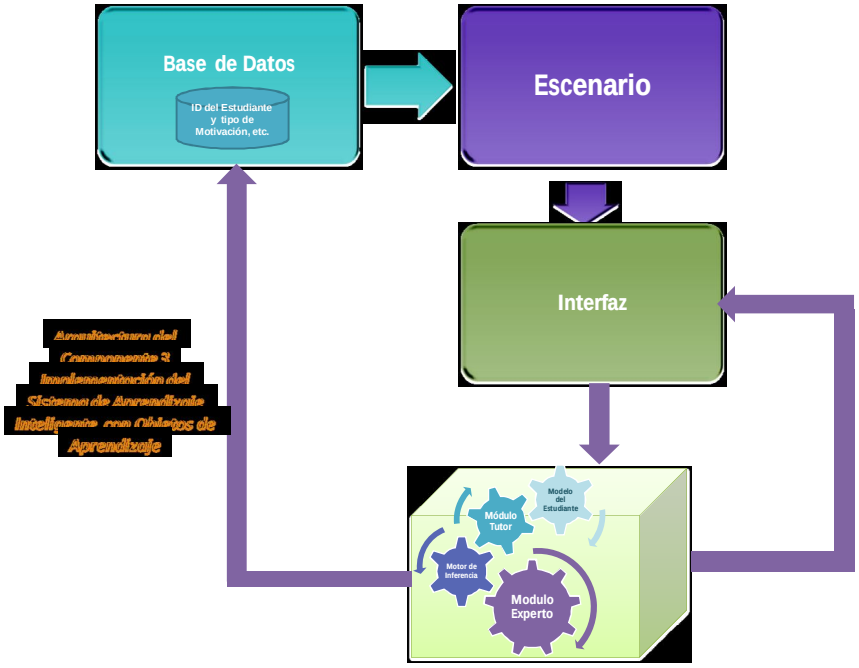


Figura 8.8 Arquitectura del Componente 3.

En los siguientes párrafos explicaremos la construcción o adaptación de cada uno de los módulos de este componente. Inicialmente nos centraremos en el diseño de los escenarios. En este diseño se tomó como referencia el grafo genético del dominio de Programación Estructurada (Ver Figura 5.1, Sección 5.2.1) y los Objetivos Instruccionales, para este trabajo.

Los escenarios y su Evaluación en el Sistema

Al evaluar un escenario se determinan los datos (valores) de los once elementos que intervienen en el proceso de enseñanza-aprendizaje. Algunos de los datos se obtienen de la evaluación de los cuestionarios (estilo de aprendizaje y motivación de estudio) y el escenario propuesto.

Para el diseño del escenario (Ver Figura 8.9 y 8.10) se programó con la herramienta Adobe Flash. Este programa es una aplicación en forma de estudio de animación que trabaja sobre "Fotogramas" destinado a la producción y entrega de contenido interactivo sin importar la plataforma. En Adobe Systems es posible utilizar gráficos vectoriales e imágenes ráster, sonido, código de programa, flujo de vídeo y audio bidireccional (el flujo de datos de subida sólo está disponible si se usa conjuntamente con *Macromedia Flash Communication Server*). En sentido estricto, Flash es el entorno y Flash

Player es el programa de máquina virtual utilizado para ejecutar los archivos generados con Flash.

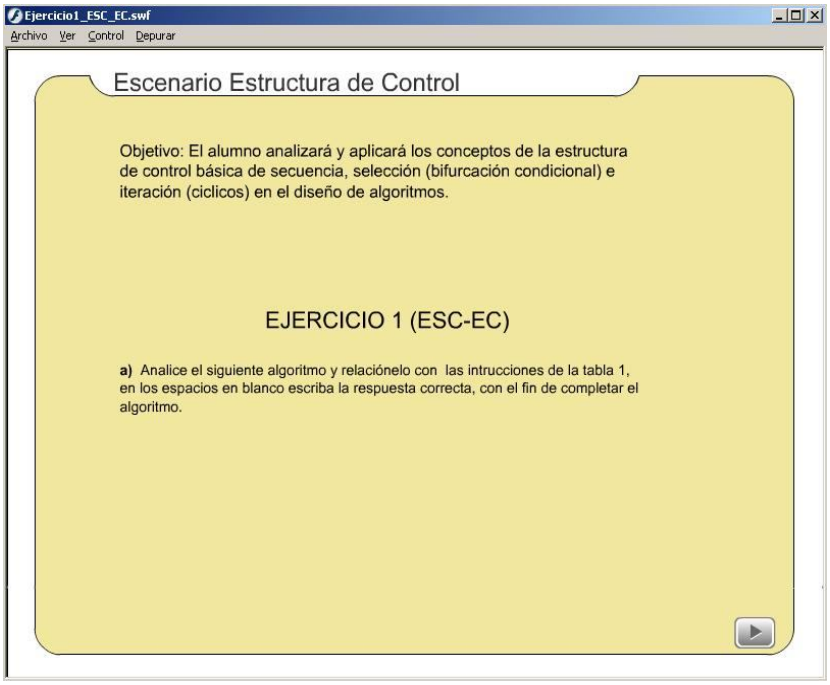


Figura 8.9 Pantalla Principal del "Escenario Estructuras de Control".

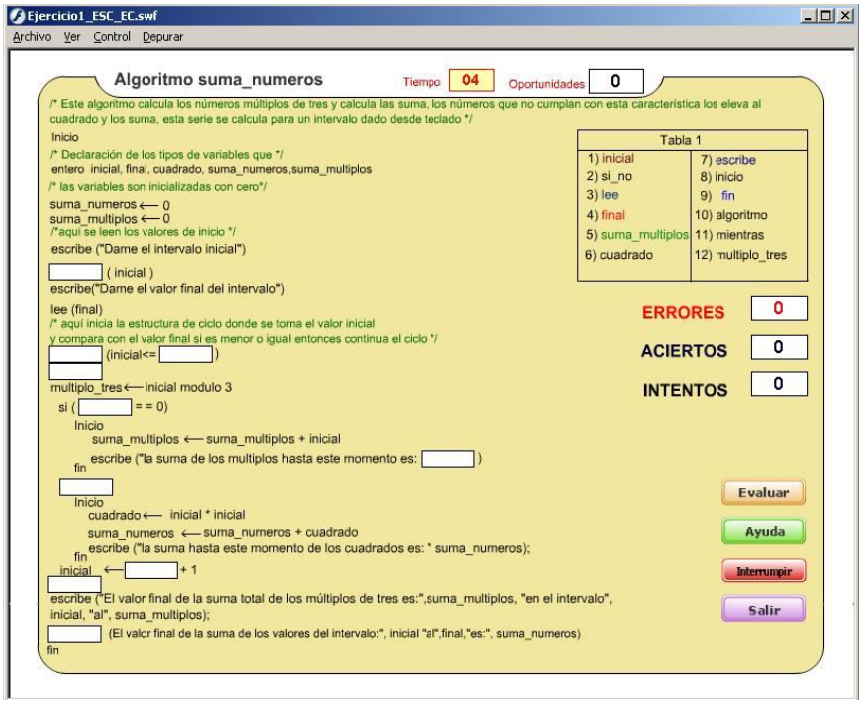


Figura 8.10 Pantalla de Presentación del Escenario Estructuras de Control.

El despliegue de la acción (llamar a una acción) se da al pulsar el botón o botones del escenarios se programó con **ActionScript** (ActionScript es un lenguaje de programación orientado a objetos (OOP), para aplicaciones web animadas dentro el entorno Adobe Flash, la tecnología de Adobe para añadir dinamismo a las presentaciones en web).

Dentro del escenario se evalúan los parámetros como: el número de errores y el tipo de error en que ha incurrido (Leve, Grave y Fatal), tiempos de inactivos, etc.

Los datos anteriores son necesarios para determinar el tipo de estrategia de enseñanza-aprendizaje a recomendar al estudiante, cabe mencionar que estos datos son guardados en la base de datos para futuras consultas.

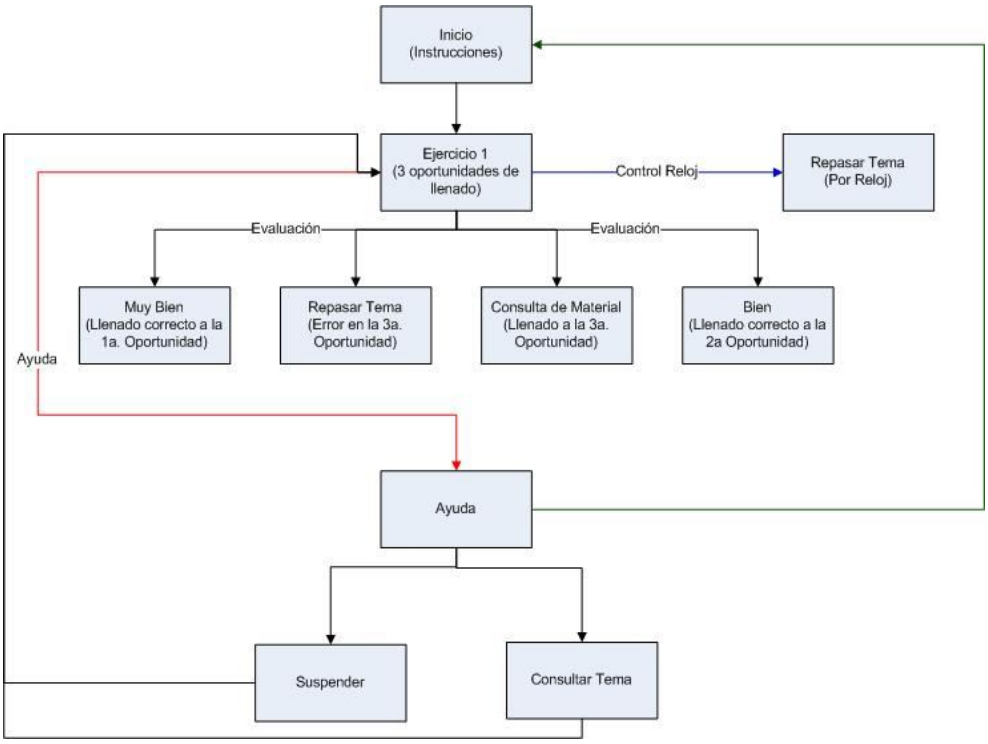


Figura 8.11 Diagrama de Bloques del Escenario 1.

En la Figura 8.11 se puede observar el Diagrama a Bloques del Escenario 1. Para mayor información de la estructura de los programas y código fuente de este módulo, consultar Anexo 3.

Al momento en que el estudiante resuelve un Escenario se generan los datos que alimentan el motor de inferencia. Estos datos conjuntamente con el resultado del cuestionario de motivación de estudio determinan las tácticas didácticas (modelo del estudiante-módulo tutor-módulo experto del SAI) a

instrumentar para que se dé el proceso de enseñanza-aprendizaje en el estudiante.

Cabe destacar que para obtener la matriz de causalidad (motor de inferencia) y los datos del vector del motor de inferencia se utilizó como herramienta la aplicación desarrollada por Laureano-Cruces, et al., (2008b).

Con la información generada anteriormente, el sistema SAI (**ProgEst**) determina qué objeto de aprendizaje se presentará al estudiante a través de la interfaz de la plataforma del LMS (por sus siglas en inglés *Learning Management System*). En este trabajo la plataforma LMS utilizada es Moodle (ver Figura 8.12). El objeto de aprendizaje presentado se encuentra en el repositorio de objetos de aprendizaje del LMS.

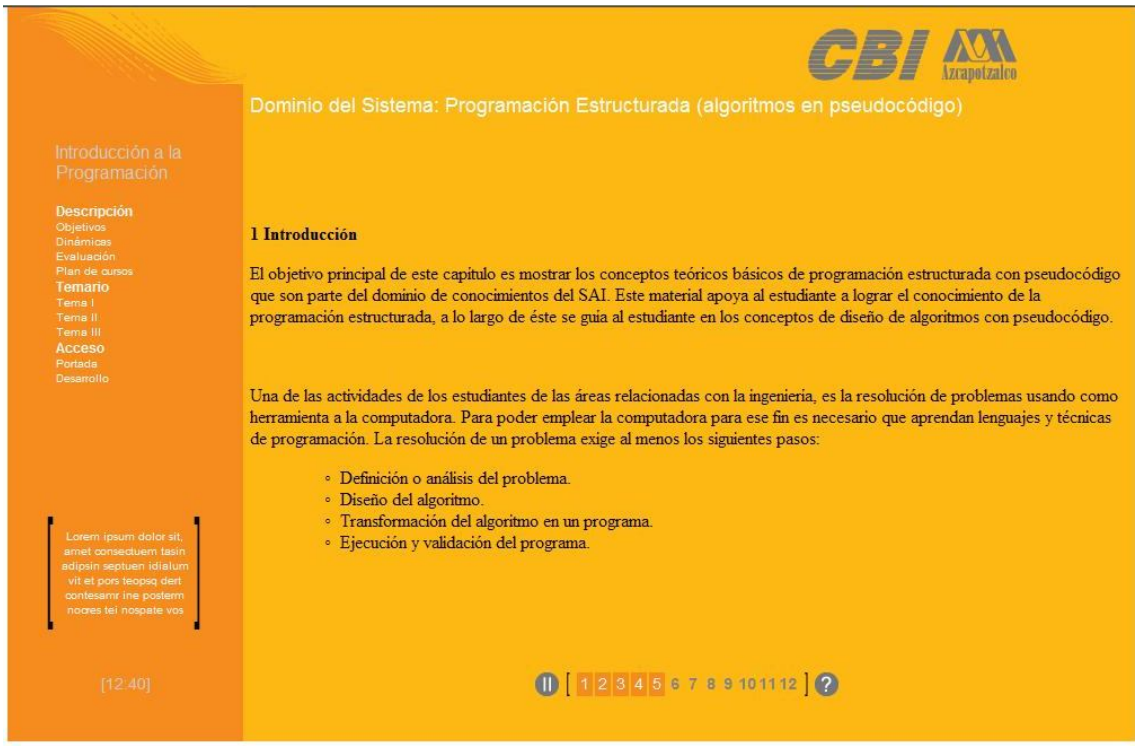


Figura 8.12 Ejemplo de la Interfaz del material del dominio "Programación Estructurada" para el estilo de aprendizaje activo.

La interfaz con la información generada anteriormente, presenta el material del dominio "*Programación Estructurada con Algoritmos en Pseudocódigo*", de acuerdo al color del estilo de aprendizaje del estudiante. (Ver Figura 8.12)

Capítulo 9 “Conclusiones y Trabajos Futuros”

En este trabajo se ha desarrollado un Sistema de Aprendizaje Inteligente con Objetos de Aprendizaje llamado “ProgEst”, aplicado al campo de la educación a distancia, vinculando las áreas del Conocimiento, las ciencias de la computación (inteligencia artificial), la pedagogía (recursos educacionales) y la psicología cognitiva (métodos de análisis de los diferentes procesos cognitivos).

La instrumentación de la Inteligencia Artificial en el Sistema le da la capacidad de recomendar estrategias en combinación del Motor de Inferencia para manejar la complejidad de predicción de tácticas didácticas para fortalecer el proceso de enseñanza-aprendizaje en el estudiante, tomando en cuenta los resultados de los cuestionarios de Estilos de Aprendizaje y de Orientación Motivacional del Estudiante.

1) En este proyecto la implementación del motor de inferencia basada en un tutor de didáctica general nos permite:

    - Una interacción dinámica en cuanto al proceso de toma de decisiones, con el fin de elegir la mejor estrategia instruccional.
    - Pronosticar el posible estado futuro, y esto nos permite personalizar las interacciones. Lo anterior a través de los once elementos que constituyen el modelo de estudiante.
    - Prevenir a través de las interacciones, estados no deseados como la renuncia o el error.
    - Nos permite un tratamiento especializado de errores por agentes reactivos.

2) Se ha creado una metodología de análisis y diseño que puede ser utilizada con el fin de crear otros SAI que cuenten con *objetos de aprendizaje*. Lo anterior a través del modelo desarrollado para el *tutor de didáctica general*.

3) Por otro lado, el hecho de que los OA cuenten con estándares de desarrollo y acceso, permitirá un mayor uso y diversificación de éstos, para las distintas implementaciones de los SAI, además de permitirnos la escalabilidad.

4) Se pretende instalar este Sistema dentro del Sistema de Aprendizaje Individualizado (SAI) de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, con el fin de potenciar el proceso de enseñanza-aprendizaje en línea.

9.1 Trabajos Futuros

Para fortalecer el proceso de enseñanza-aprendizaje se tiene considerado:

- Incluir más sub-tutores especializados.
- Continuar con el desarrollo de la interfaz, implementando un agente pedagógico.
- Dado al amplio campo de aplicación de este sistema, es posible adecuarlo a otros dominios de conocimiento.
- Desarrollar material didáctico de acuerdo al estilo de aprendizaje.

Referencias

ADL, (2002). *Emerging and Enabling Technologies for the design of Learning Object Repositories Report. Advanced Distributed Learning Initiative*. Obtenido en: http://xml.coverpages.org/ADLRepositoryTIR.pdf.

Aguilar Cisneros J., Muñoz Arteaga J., Pomares Hernández S., (2004). *Guías de diseño para el desarrollo de objetos de aprendizaje*. Workshop Learning Objects Technology, 5th Mexican International Conference on Computer Science (ENC 2004), Colima, Colima.

Alonso M. C., Gallego D. J. y Honey P., (1994). *Los Estilos de Aprendizaje (Procedimientos de Diagnóstico y Mejora)*. España: Ediciones Mensajero S.A. Sexta Edición, ISBN 84-271-1914-3.

Arruarte A., (1998). *Fundamentos y diseño de IRIS. Un entorno para la generación de Sistemas de Enseñanza Inteligentes*. Tesis Doctoral (Tercer Ciclo) de la Universidad del País Vasco, San Sebastián.

ASTD y SmartForce, (2003). *A field guide to learning object*. Retrivied.

Azpiazu J., Pazos J. y Silva A., (2001). *La teleformación mediante Internet*. España, en: Actas de "El futuro de Internet. Acceso y Teleservicios", Fundación Alfredo Brañas.

Berners-Lee T., (2000). *Tejiendo la red: El inventor del World Wide Web nos descubre su origen*. Madrid: Siglo XXI.

Bloom, B.S. (Ed.) (1956). *Taxonomy of educational objectives, the classification of educational goals - Handbook I: Cognitive Domain*. New York: McKay.

Casas Armengol M. y Pomerhanz Stojanovic L., (2008). *Tendencias e Innovaciones en Universidades a Distancia*. Artículo del Libro: El futuro de la educación a distancia y del e-learning en América Latina (Una Visión Prospectiva). D.F., México: Instituto Latinoamericano de la Comunicación Educativa (ILCE).

Castañeda Figueiras S., (). *Educación, aprendizaje y cognición (Teoría en la Práctica)*. D.F., México: El Manual Moderno. Facultad de Psicología, Universidad Nacional Autónoma de México.

Chan Núñez M. E., Galeana de la O L., Ramírez Montoya M. S., (2006). *Objetos de Aprendizaje e Innovación Educativa*. México: Editorial Trillas, p.p. 11-44.

Dato Abstracto (definición). Obtenido en: http://es.wikipedia.org/wiki/Tipo_de_dato_abstracto.

De Antonio A., Barreiro J. M., Crespo J., Pazos J., Rodríguez Patón A. y Silva A., (2003). *Informe técnico sobre e-learning para la Agencia Tributaria Española*. Madrid, España.

De Arriaga Gómez F., El Alami M., Laureano Cruces A., Martínez Ugena A. y Arriaga Perelló A., (2002). *E-Learning: Una nueva Generación de Sistemas Inteligentes de Aprendizaje*. España: Revista Fundatel (Revista Informativa), Número 9, pp. 20-28.

De Kereki Guerrero F. I., (2003). *Modelo para la Creación de Entornos de Aprendizaje basados en Técnicas de Gestión del Conocimiento*. Tesis Doctoral de la Universidad Politécnica de Madrid. Obtenido desde: www.ort.edu.uy/fi/pdf/Tesis.pdf.

Eiseman L., (2006). *Color-A PANTONE Color Resourse. Editorial: Hand Books Press*. ISBN-13: 978-0971401068,

Escorcia Saldariaga G., (2006). *E-learning sin límites*. Boletín Cuatrimestral del Centro Interactivo Multitecnológico de Educación a Distancia (CIMED). D.F., México: Instituto Latinoamericano de la Comunicación Educativa, 22-24 p.p.

Fernández I., (1989). *Estrategias de Enseñanza en un Sistema Inteligente de Enseñanza asistida por Ordenador*. Tesis Doctoral (Tercer Ciclo) de la Universidad del País Vasco, San Sebastián.

Girard J., Gauthier G. & Levesque S., (1992). *Une Architecture Multiagent*. En Memorias Second International Conference, ITS'92. Lectures Notes in Computer Science No. 608. Springer-Verlag, pp. 172- 182.

Ginsburg H. & Opper S., (1986). *Piaget y la Teoría del Desarrollo Intelectual*. Prentice Hall.

Goldstein I., (1976). *The Computer as a Coach: An Athletic Paradigm for Intellectual Education*. Memo No. 389, December, MIT Artificial Intelligence Laboratory.

Goldstein I., (1979). *The Genetic Graph: A Representation for the Evolution of Procedural Knowledge*. Man Machine Studies, 11, 51-77.

Goñi J. J., (2000). *El nuevo entorno de aprendizaje: un organizador de los componentes del teleaprendizaje a través de redes telemáticas*. Obtenido desde http://encina.cnice.mecd.es/~fvab0001/textoaprendizaje.html.

Gutiérrez J., (1994): *INTZA: Un sistema tutor inteligente para entornos industriales*. Tesis doctoral. Universidad del País Vasco / Euskal Herriko Unibertsitatea (UPV-EHU): San Sebastián, Gipuzkoa (España).

Hatala M., Richards G., Eap. T. & Willms J., (2004). *Interoperabiliy of Learning Object Repositories and Services: Standards, Implementations and Lessons Learned*. Proceedings of the 13 th WWW Conference. Obtenido en: http://www.www2004.org/proceedings/docs/2p19.pdf.

Islas E., (2001). *Ideas para reflexionar en torno a la importancia y necesidad de implementar una asignatura relativa a la formación de usuarios de la información*. Obtenido desde www.uqroo.mx/uqroo/eventos/textocursos.html.

Jacques P. & Viccari R., (2004a). *A BDI Approach to Infer Student's Emotions*. IBERAMIA2004, Lecture Notes on Artificial Intelligence, 3315: 901-911.

Jacques P., (2004b). *Using an Animated Pedagogical Agent to Interact Affectively with the Student*. Ph.D. Thesis. Universidade Federal do Rio Grande do Sul- Instituto de informática.Consulted from http://www.inf.ufrgs.br/~pJacques/

JISC, (2000). Joint Information Systems Committee "Requirements for a Virtual Learning Environment". Obtenido desde http://www.jisc.ac.uk/index.cfm?name=mle_related_vle.

Joyanes L., (1996). *Fundamentos de Programación "Algoritmos y Estructura de Datos*". Distrito Federal, México: Mc. Graw Hill, Segunda Edición.

Ketudat S., (2000). *Priorities for a dynamic university system: Thailand En: The universities responsibilites to society. International Perspectives*. Holanda: Elsevier Sciencia Ltd, Editor: Guy Neave.

Krug S., (2006). *No Me Hagas Pensar*. Madrid, España: Pearson Educación.

Kruse K., (2006). *Introduction to Instructional Design and the ADDIE Model*. Obtenido en: http://wwww.e-learningguru.com/articles/art2_1.html.

L'Allier J. J., (1997). *Frame of Reference: NETG´S Map to Its Products, Their Structures and Core Beliefs*.

Lamarca M., (2006). *Hipertexto: El nuevo concepto de documento en la cultura de la imagen*. Tesis Doctoral, Universidad Complutense de Madrid. España. Obtenido en: http://www.hipertexto.info/documentos/interfaz.htm.

Landeta Etxeberría A., (2007). *Libro de Buenas Prácticas de e-learning*. Capítulo 16 "*Estándares de e-learning*". Madrid, España. Universidad a Distancia de Madrid. Obtenido en: http://www.buenaspracticas-elearning.com.

Laureano Cruces A., (1997). *Multiagentes en el Diseño de un Sistema de Enseñanza Inteligente*. Perfiles Educativos, 75, 23-33.

Laureano Cruces A., (2000a). *Interacción Dinámica en Sistemas de Enseñanza Inteligentes*. Tesis Doctoral. Instituto de Investigaciones Biomédicas/UNAM. Obtenido en: http://delfosis.uam.mx/~ana/02_publicaciones/tesis_doctoral/TesisDoctoral.PDF .

Laureano Cruces A. & De Arriaga F., (2000b). *Reactive Agent Design for Intelligent Tutoring Systems*. En *Cybernetics and Systems. An International Journal.*

Laureano Cruces A., Terán Gilmore A., De Arriaga Gómez F. & El Alami M., (2003). *La Importancia de las Estrategias Cognitivas en el Diseño del Currícula Didáctico*. Vol. I, pp. 35–41. En XVI Congreso Nacional y II Congreso Internacional de Informática y Computación de la ANIEI. 22-24 de octubre de 2003, Zacatecas, México.

Laureano Cruces A., Terán Gilmore A. & De Arriaga Gómez F., (2004a). *A Learning Model Based on a Didactic Cognitive Approach: The Case Of Single-Degree-Of-Freedom Systems. Computer Applications in Engineering Education*. En Computer Applications In Engineering Education. ISSN: 1061-3773, John Wiley and Sons. Inc. USA. Volume 12, Number 3, pp. 152-164.

Laureano Cruces A., De Arriaga Gómez F. & El Alami M., (2004b). *Técnicas Útiles a la Construcción de Simulación Inteligente en el Campo Educativo*. Autores del Capítulo 36 del libro '*Educación, aprendizaje y cognición. Teoría en la Práctica*'. Publicado por la Universidad de Guadalajara y la Editorial El Manual Moderno, pp. 557-574.

Laureano Cruces A., Ramírez Rodríguez J., De Arriaga Gómez F., & Escarela Pérez R., (2006). *Agents Control in Intelligent Learning Systems: The Case of Reactive Characteristics*. Reino Unido: Revista Interactive Learning Environments 14(2), pp. 95-118. ISSN: 1049 - 4820, Taylor and Francis. (Revista indizada en Science Citation Index).

Laureano Cruces A., Sánchez Guerrero L., Ramírez Rodríguez J. & Mora Torres M., (2008a). *Learning Objects and Personalized Instruction*. In G. Richards (Ed.), *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2008*, (pp. 1728-1736). Chesapeake, VA: AACE. ISBN: 1-880094-66-5.

Laureano Cruces A., Mora Torres M., De Arriaga Gómez F., Ramírez Rodríguez J. & Escarela Pérez R., (2008b). Cognitive-Operative Model of Intelligent Learning Systems Behavior. Interactive Learning Environments. ISSN: 0888-613X, Elsevier. UK. Publicado en Línea, http://www.informaworld.com/smpp/content~content=a795273447~db=all~order=pubdate.

Laureano Cruces A., Mora Torres M., Ramírez Rodríguez J. & Gamboa Rodríguez F., (2009). *Emotions as an Element that Maximizes the Effectiveness of a Pedagogical Agent*, aceptado en World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (E-Learn 2009).

Levine G., (1990). *Introducción a la Computación y a la Programación Estructurada*. Distrito Federal, México: Mc. Graw Hill, Segunda Edición.

López C., (2005). *Los Repositorios de Objetos de Aprendizaje como soporte a un entorno e-learning*. Tesina Doctoral, Universidad de Salamanca. (Director Francisco José García Peñalvo). Obtenido en: http://www.biblioweb.dgsca.unam.mx/libros/repositorios/index.htm.

Lucas Marín A., (2000). *La nueva sociedad de la información*. España: Editorial Trotta.

Marchesi Á. y Martín E., (1998). *Calidad de la enseñanza en tiempos de cambio*. España: Alianza Editorial SA.

Marina T., Feixas M. y Marqués P., (1999). *La universidad ante los retos que plantea la sociedad de la información. El papel de las TIC*. Sevilla, España: Actas de las Jornadas EDUTEC-99. Obtenido en: http://tecnologiaedu.us.es/edutec/paginas/p2f2.htm.

Massart D. & Dung L. T., (2004). *Federated Search of Learning Object Repositories: The CeLeBraTe Approach*. International Conference RIVF´04 (pp. 143-146).

Mayorga R., (1999). *Los desafíos a la universidad latinoamericana en el siglo XXI*. España: Revista Iberoamericana de Educación. Número 21, Septiembre-Diciembre 1999, (pp. 25 a 40).

Miklos T. y Arroyo García M., (2006).*Bitácora del porvenir: primera aproximación, "El Futuro de la Educación a Distancia y del E-Learning en América Latina"* .Boletín Cuatrimestral del Centro Interactivo Multitecnológico de Educación a Distancia (CIMED). D.F., México: Instituto Latinoamericano de la Comunicación Educativa, 25-27 p.p.

Miklos T. y Arroyo M., (2008). *El futuro de la educación a distancia y del e-learning en América Latina (Una Visión Prospectiva)*. D.F., México: Instituto Latinoamericano de la Comunicación Educativa (ILCE).

Morales E., García Francisco J., Barrón A., Berlanga A. J. y López C., (2005). Propuesta de Evaluación de Objetos de Aprendizaje. Instituto Universitario de

Ciencias de la Educación, Universidad de Salamanca, 37008. Salamanca España.

Muñoz Arteaga J. et al., (2005). *Reuniones de trabajo nacionales a través de diversos foros nacionales*.

Muñoz Arteaga J., Osorio Urrutia B., Álvarez Rodríguez Francisco J., Cardona Salas P., (2006). *Metodología para elaborar objetos de aprendizaje e integrarlos a un Sistema de Gestión de Aprendizaje*. D.F., México: Universidad Autónoma de Aguascalientes y Universidad de Guadalajara Virtual.

Muñoz Arteaga J., Álvarez Rodríguez F. J., Chan Núñez M. E., (2007). *Tecnología de Objetos de Aprendizaj*e. D.F., México: Universidad Autónoma de Aguascalientes y Universidad de Guadalajara Virtual.

Pazos Sierra J., (2001). *Enseñanza del futuro: a grandes males pequeños remedios*. España: Universidad Politécnica de Madrid.

Pintrich P., Smith D., García T. y Mckeachie W., (1991). *A Manual for the Use of the Motivated Strategies for Learning Questionarie (MSQL)*. Michigan, USA: National Center for Research to Improve Postsecondary Teaching and Learning University of Michigan.

Rama C., (2006). *La Tercera Reforma de la Educación Superior en América Latina y el Caribe: masificación, regulaciones e internalización*. Informe sobre la Educación Superior en América Latina y el Caribe 2000-2005. Caracas, Venezuela: UNESCO/IESALC.

Ramírez Rodríguez J., (1994). *La Teoría de las Gráficas*. En Línea 1(4), 3-4.

Reilly R. R. & Lewis E. L., (1991). *Educational Psychology Application for Classroom, Learning and Introduction*. Editions. Maxwell McMillan International.

Rodríguez Aguilar R. M., (2006). *Tutor de Didáctica General*. MSc. Degree Thesis. Posgrado en Ciencia e Ingeniería de la Computación-Universidad Nacional Autónoma de México.

Salinas J., (1995). *Cambios en la comunicación, cambios en la educación*. Sevilla: Universidad de Sevilla. Obtenido en: http://www.uib.es/depart/gte/cambios.html.

Sánchez Guerrero L., Laureano Cruces A., Ramírez Rodríguez J. & Mora Torres M., (2009). *An Intelligent Learning System within a Learning Object*, aceptado en World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (E-Learn 2009).

Sánchez Guerrero L., (2001). *Curso de Introducción a la Programación*. Obtenido en: http://luda.uam.mx/.

SCORM ADL, (2005). *Shareable Content Object Reference Model*. Obtenido en: http://www.adlnet.org/scorm/index.cfm.

Sutton T. y Whelan B., (2006). *La Armonía de los Colores*. Editorial Blume. ISBN 9788480766722

Thropp S., (2004a). *Shareable Content Object Reference Model 2004: Overview. 2a. edición. Documentation Suite*. Obtenido en: http://www.adlnet.org/scorm/index.cfm.

Thropp S., (2004b). *SCORM Content Aggregation Model Versión 1.3.1. Documentation Suite*. Obtenido en: http://www.adlnet.org/scorm/index.cfm.

Thropp S., (2004c). *SCORM Run Time Environment Versión 1.3.1. Documentation Suite*. Obtenido en: http://www.adlnet.org/scorm/index.cfm.

Thropp S., (2004d). *SCORM Sequencing and Navigation. Documentation Suite*. Obtenido en: http://www.adlnet.org/scorm/index.cfm.

Velasco Santos P., Laureano Cruces A., Mora Torres M. y Sánchez Guerrero L., (2008). *La Importancia del Diseño de una Interfaz en el Proceso de Enseñanza-Aprendizaje*. Memorias en CD, ISBN 978-970-15-14388-2, pp. 108-113. En el XXI Congreso Nacional y VII Congreso Internacional de Informática y Computación de la ANIEI. 1-3 de octubre, Monterrey.

Velasco Santos P., Laureano Cruces A., Mora Torres M. y Sánchez Guerrero L., (2009). *Un Diseño de Interfaz tomando en cuenta el Estilo de Aprendizaje*. Sometido al XXII Congreso Nacional y VIII Congreso Internacional de Informática y Computación de la ANIEI.

Wiley D. A., (2000). *Connecting learning objects to instructional design theory: a definition, a metaphor, and a taxonomy*. En D.A. Wiley (Ed.): The Instructional Use of Learning Objects. Obtenido en: http://reusability .org/read/chapters/wiley.doc.

Wilson B., (1995). *Metaphors for instruction: why we talk about learning environments*. En: Educational Technology. Volumen 35, No. 5, p. 25-30.

Anexo 1 "Metodología de Diseño de Objetos de Aprendizaje"

Introducción

El propósito de este anexo *es presentar la metodología propuesta por* (Muñoz Arteaga J. et al., 2006), para el proceso de diseño e implementación de los objetos de aprendizaje (OA). Para este trabajo de tesis se consideró importante seguir paso a paso la metodología en el diseño de los Objetos de Aprendizaje que constituyen el Dominio de Conocimiento "Programación Estructurada con Algoritmos en Pseudocódigo".

A1.1 Metodología de Diseño de los OA

En esta sección nos centraremos en la metodología de elaboración de OA. Es importante mencionar que la metodología que se describirá en esta sección se tomo como referencia el trabajo desarrollado por (Muñoz Arteaga J. et al., 2006) y el modelo ADDIE (Kruse K., 2006), el cual es el modelo apropiado para el análisis, diseño, desarrollo, evaluación, implantación y evaluación de materiales y actividades de aprendizaje. A partir del modelo ADDIE, se llega a la metodología ADDIEOA (análisis, diseño, desarrollo, evaluación e implantación de un objeto de aprendizaje), compuesta por 5 fases. (Ver Figura A.1)

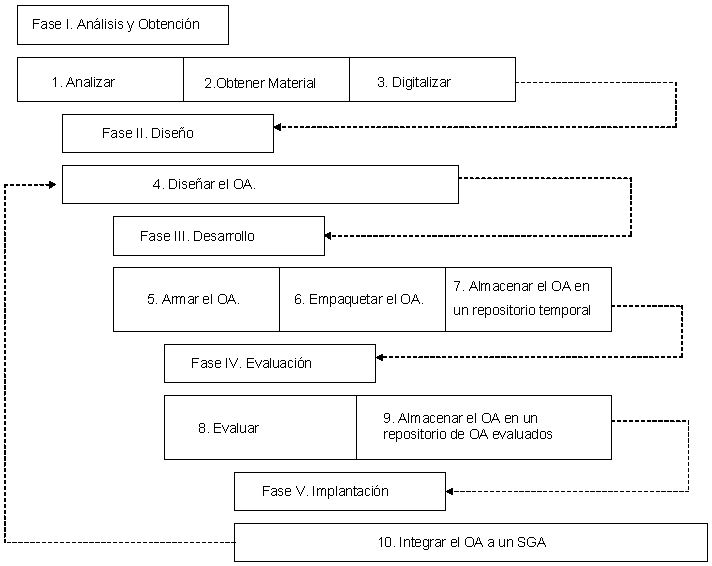


Figura A1.1 Fases de la Metodología Propuesta. (Muñoz Arteaga J., Osorio Urrutia B., Álvarez Rodríguez Francisco J., Cardona Salas P., 2006)

Como se puede observar en la Figura A.1, cada fase está formada por actividades específicas, las cuales son responsabilidad de diferentes actores, cada fase logra como resultado “un producto final”, como se observa en la Figura A.2.

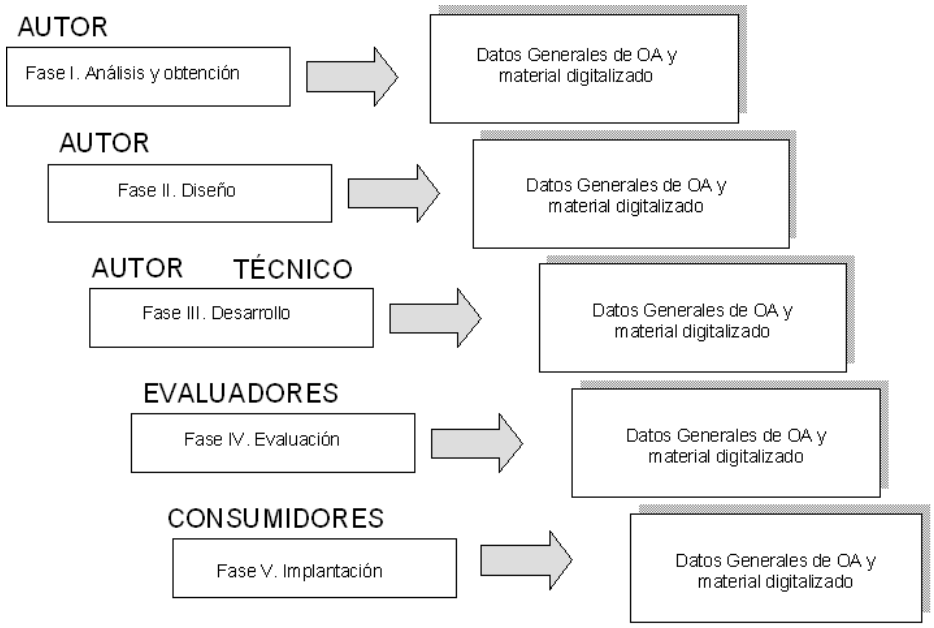


Figura A1.2 Productos Terminales y Actores de cada Fase. (Muñoz Arteaga J., Osorio Urrutia B., Álvarez Rodríguez Francisco J., Cardona Salas P., 2006)

Es importante mencionar a los actores que intervienen en el proceso de cada una de las fases:

- Autores (docentes): Generadores y propietarios de los contenidos informativos, para aprender a partir de los mismos o generar nuevos.
- Técnico (técnico en computación): Programadores y conocedores del software propio para el empaquetamiento, almacenamiento y difusión de un OA.
- Grupo de experto (evaluadores): Este grupo está integrado por docentes, técnicos y pedagogos, los cuales se encargarán de la evaluación del OA.

Las cinco fases que conforman la metodología propuesta son:

***Fase 1. Análisis y Obtención*:** se identifica una necesidad de aprendizaje (resolver un problema, mejorar, innovar), con base en esto se tiene claro que se va a enseñar, se identifican los datos generales del OA, y se obtiene el material didáctico necesario para realizarlo. En esta fase interviene directamente el autor.

Fase 2. *Diseño*: Es importante dejar claro cómo se va a enseñar, para esto es necesario desarrollar un esquema general del OA, el cual indicará cómo están interrelacionados los contenidos, objetivos, actividades de aprendizaje y la evaluación. Es importante considerar en esta etapa el metadato, el cual es información general del OA, más adelante hablaremos a detalle sobre el mismo. En esta fase interviene el autor.

Las Fases 1 y 2 son las apropiadas para definir “de forma clara” la parte pedagógica del OA; de un objetivo de aprendizaje bien planteado se derivarán los contenidos informativos, actividades y evaluaciones necesario para adquirir un determinado aprendizaje.

Fase 3. Desarrollo: Mediante un software generador de código html ó xml, ó directamente en cada uno de estos lenguajes se armará la estructura del esquema general del OA elaborado en la fase de diseño. En esta fase es importante que intervenga el técnico.

Fase 4. Evaluación: Hay una serie de factores a evaluar en un OA, que van desde el diseño hasta el aspecto pedagógico, se considera necesario diseñar un instrumento de evaluación que considere cada factor a evaluar, así como los indicadores de los mismos.

Fase 5. Implantación: El OA será integrado en un sistema de gestión de aprendizaje, el cual puede ser propio o comercial. Esto es con la finalidad de hacer uso y rehúso, e interactuar con él en un determinado contexto. Esta fase será la pauta para que el OA sea evaluado por los usuarios, los cuales pueden proveer una retroalimentación valiosa.

Cada una de las fases anteriores está integrada por actividades que permiten generar un producto final en cada una de ellas.

Es importante aclarar que durante el transcurso de la metodología propuesta se recomienda utilizar algunas plantillas, esto con la finalidad de facilitar la obtención de información por parte de los docentes. Estas plantillas de apoyo al igual que la metodología de diseño de los OA son resultado de los trabajos de (Muñoz Arteaga J. et al., 2006). A continuación se explica a detalle cada una de las fases mencionadas con anterioridad:

A1.1.1 Fase 1 "Análisis y Obtención"

En esta fase se identificarán los datos generales del OA, se recolectará y, si es necesario, se digitalizará el material informativo necesario para su realización. (Ver Figura A.3)

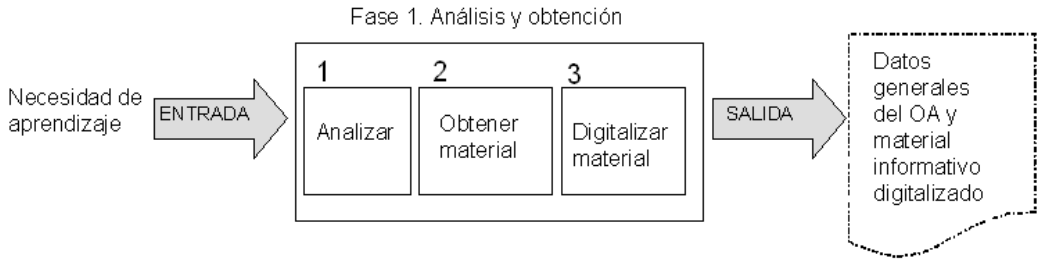


Figura A1.3 Fase 1 "Análisis y Obtención de Información". (Muñoz Arteaga J., Osorio Urrutia B., Álvarez Rodríguez Francisco J., Cardona Salas P., 2006)

Paso 1 "Analizar"

En este paso es importante dejar claro qué es lo que se requiere enseñar, es decir, que se identifiquen los datos generales del OA. Esto resulta más fácil al llenar la plantilla Análisis. (Ver Tabla A.1)

Tabla A1.1 Plantilla de Análisis.

Análisis	
Nombre del OA	El nombre del objeto de aprendizaje deberá representar de forma clara y simple el contenido tratado, evitando la ambigüedad en la idea. Por ejemplo derecho civil, inteligencia artificial, formato de fuente, etcétera.
Descripción del OA.	Descripción textual del contenido del OA.
Nivel escolar al que va dirigido el OA.	Contexto principal en el que será usado el OA. Por ejemplo: Primaria, secundaria, bachillerato, licenciatura y postgrado.
Perfil del alumno al cual va dirigido el OA (necesidad de aprendizaje)	Este perfil está íntimamente ligado con el perfil requerido en el curso en cual será utilizado el OA.
Granularidad	Responde al tamaño de los OA. Mientras más pequeños sean aumenta su capacidad de reutilización en otros contextos. Ejemplificando, tenemos: 1) Tema, 2) unidad, 3) Materia, 4) Plan de Estudios. Siendo 1 la granularidad más pequeña.

Paso 2 "Obtener el Material"

Este paso consiste en proveer el material didáctico necesario para la construcción del OA, el cual puede ser de diversa índole, como por ejemplo: impresos (textos) –libros, enciclopedias, fotocopias, periódicos, documentos-, imágenes fijas proyectables (fotos) –diapositivas, fotografías-, materiales sonoros (audio) –cintas, discos, programas de radio-, materiales audiovisuales (video) – montajes audiovisuales, películas, videos, programas de televisión- materiales electrónicos –Internet, discos compactos-. Para una mejor organización en la obtención del material didáctico se sugiere llenar la plantilla Obtención. (Ver Tabla A.2)

Tabla A1.2 Plantilla de Recolección.

Obtención	
Tipo de material	Fuente
Impresos (textos): Libros	Manual de Office XP
Texto electrónico	Ayuda de Microsoft Word

Al llenar la plantilla anterior, permite tener cada una de las referencias del contenido informativo utilizado en la elaboración del OA, y evita problemas de derechos de autor.

Paso 3 "Digitalizar el Material"

Este paso sólo es necesario en el caso de que el material informativo no esté previamente digitalizado. El material puede descargarse de Internet o de las fuentes, o en su defecto digitalizarse, es decir, capturar el texto dentro de un procesador, las fotografías o videos mediante cámara digital y las imágenes con un escáner. Recuerde hacer la referencia apropiada de cada uno de los materiales digitalizados, para evitar problemas de derechos de autor.

El producto que se obtiene al finalizar esta fase es el conjunto de datos generales del OA, así como el material didáctico digitalizado. Es importante mencionar que es indispensable identificar cada uno de los componentes para realizar la estructura del objeto de aprendizaje.

A1.1.2 Fase 2 "Diseño"

En esta fase se identificarán cada uno de los componentes del OA, así como su interrelación, hasta concluir con su estructura general. Hay que aclarar que esta fase es una de las más importantes, ya que en ésta se específica la parte pedagógica del OA, en la cual se tiene que poner especial atención. (Ver Figura A.4)

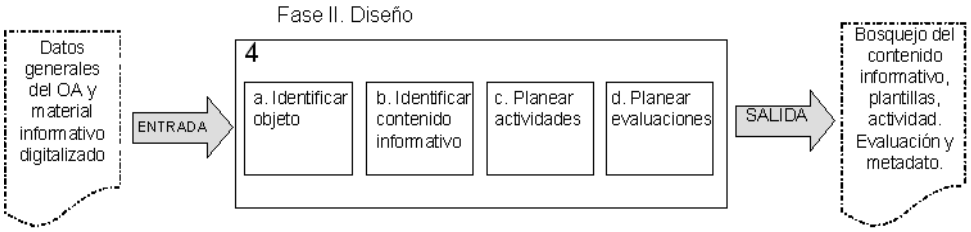


Figura A1.4 Fase 2 "Diseño". (Muñoz Arteaga J., Osorio Urrutia B., Álvarez Rodríguez Francisco J., Cardona Salas P., 2006)

Paso 4 "Diseñar el OA"

Es fundamental identificar cada uno de los componentes del OA.

a) **Objetivo.** Recuerde considerar el objetivo de aprendizaje planteado en la fase de análisis. Si aún no lo tiene claro es momento de hacerlo. Este objetivo debe ser redactado en términos de lo que el alumno aprenderá o sabrá hacer al final de la interacción con el OA.

b) **Contenido Informativo.** Es recomendable hacer uso de múltiples recursos digitales como textos, imágenes, videos, animaciones, etcétera (digitalizados en la Fase 1). Éstos deben ser organizados de una forma adecuada para que se capte la atención del alumno y se facilite su aprendizaje. En este punto se sugiere que el docente realice un esquema del acomodo del contenido informativo. En la Figura A.5 se muestra un ejemplo.

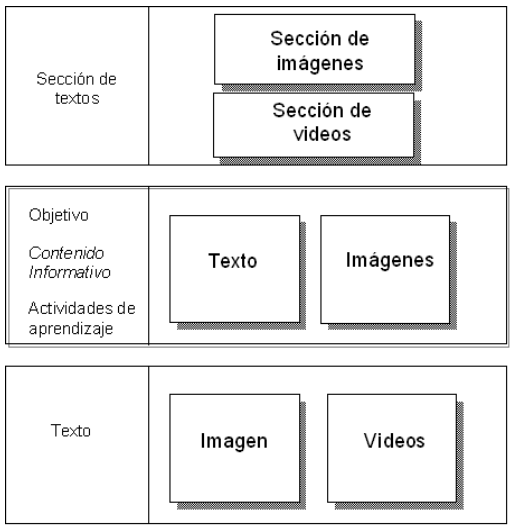


Figura A1.5 Acomodo sugerido del Contenido Informativo. (Muñoz Arteaga J., Osorio Urrutia B., Álvarez Rodríguez Francisco J., Cardona Salas P., 2006)

c) **Actividades.** Conjunto de pasos y etapas que el estudiante realizará con el fin de promover y facilitar su proceso de aprendizaje. Algunos ejemplos de actividades son: lecturas, resúmenes, realización de ejercicios, simulaciones, etc. Valore que las actividades propuestas estén relacionadas con su objetivo de aprendizaje. Se sugiere llenar la plantilla Actividad, en la que se especifiquen el número de actividad, propósito, descripción y tipo de archivo en el cual está la actividad. Si es una actividad de línea, como por ejemplo foros o Chat, sólo especifique "en línea". (Ver Tabla A.3)

Tabla A1.3 Plantilla de Actividades.

No.	Propósito de la Actividad	Descripción de la Actividad	Tipo de Archivo
1.	Que el alumno aplique formato de fuente a un texto.	Aplique negrita, cursiva sólo a los títulos del documento anexo.	Word
2.	Que el alumno aplique formato de párrafo a un texto.	Aplique la alineación justificada e interlineado doble al documento anexo.	Word

Las actividades de foros y Chat, por lo regular son parte del sistema de gestión de aprendizaje, donde serán integrados los OA. Pero no es limitante para que un OA pueda integrarlos.

d) **Evaluación del aprendizaje.** Uno de los objetivos de los OA es asegurar que el objetivo de aprendizaje planteado se alcance, por lo anterior es importante implementar actividades que evalúen los conocimientos. Programe alternativas para evaluar, por ejemplo: cuestionarios en línea, productos significativos, reportes de lecturas, etc. Se sugiere para mantener un mejor control del aprendizaje dentro del OA, implementar evaluaciones en línea, necesariamente cada uno de éstos deberá mostrar al alumno la respuesta correcta, una vez contestada la pregunta; de igual forma, al finalizar el test, éste debe presentar el listado de preguntas buenas y malas, así como el puntaje final alcanzado por el alumno. Se sugiere llenar la plantilla de Evaluación, la cual indicará el número de evaluaciones que se realizarán al finalizar la interacción con el OA, número y tipo de preguntas. (Ver la Tabla A.4)

Tabla A1.4 Plantilla de Evaluación.

Evaluación		
No.	**No. de preguntas**	**Tipo**
1.	10	Falso/verdadero
2.	5	Opción Múltiple

e) Metadato. Es información acerca de la información, describe al OA por ejemplo: nombre, autor, nivel escolar al que va dirigido, nivel taxonómico.

Gracias a la información del metadato se facilita la gestión de los OA para su catalogación, búsqueda y recuperación. Existen varios estándares para llenar de forma correcta el metadato. En esta metodología se utilizará el estándar SCORM. El metadato de SCORM se basa en el estándar IEEE 1484.12.1-2002 LTSC* Learning Object Meta-data (LOM), SCORM especifica las características generales del metadato de un OA: Éste está organizado en nueve categorías, cada una de ellas contiene un número determinado de elementos. Para efectos de la metodología sólo será necesario llenar cuatro de las nueve categorías.

Se seleccionaron estas cuatro categorías por ser las que describen los datos generales, pedagógicos y de derecho de autor del OA, el actor será el encargado del llenado de las mismas, las otras cinco categorías son cuestiones técnicas del OA, y serán llenadas por los técnicos.

Las cuatro categorías seleccionadas son las siguientes:

- General.
- Ciclo de vida.
- Educacional.
- Derechos.

Para facilitar la identificación de cada uno de los elementos de la categoría se sugiere llenar las plantillas: categoría general, categoría ciclo de vida, categoría educacional y categoría derechos de propiedad. Todas estas plantillas a su vez formarán la plantilla Metadato. (Ver Tablas A.5-A.8)

Cabe aclarar que las categorías, así como sus elementos fueron tomadas textualmente, sólo en el caso de los elementos cobertura y entrada se adaptaron a necesidades específicas del contexto en el que se aplicará la metodología.

Tabla A1.5 Plantilla de la Categoría General.

Categoría General. Agrupa la información de carácter general que identifica al OA.	
Identificador	No será llenado.
Título	Nombre dado al OA.
Entrada del Catálogo	
Catálogo	No será llenado.
Entrada	Calificación dada al OA por los evaluadores.
Lengua	Idioma(s) utilizados en el OA, para interacción con el usuario. Se seleccionará de una lista de opciones.
Descripción	Descripción textual del contenido de este OA.
Descriptor (palabra clave)	Palabra(s) clave o frase que describa al OA,
Cobertura	Para este caso se procederá a llenar con la categoría del OA, hasta el momento cuenta con las siguientes: 1. Morfología 2. Ingeniería de Software 3. IHC 4. Estructura de datos 5. Lenguaje de programación 6. Tecnologías de información 7. Biología 8. Estadística 9. Filosofía
Escritura	Jerarquía o estructura organizativa del OA. Atómico: un objeto que es indivisible (en ese contexto). Colección: un conjunto de objetos relacionados, cuyas relaciones no están especificadas. Jerárquica: un conjunto de objetos cuyas relaciones pueden representarse por un árbol jerárquico. Lineal: Un conjunto de objetos que tienen un orden específico.
Nivel de agregación	Granularidad del OA. 1= Tema 2= Unidad 3= Curso 4= Plan de estudios

Tabla A1.6 Plantilla de la Categoría de Ciclo de Vida.

Categoría ciclo de vida. Agrupa las características relacionadas con la historia y el estado actual del OA, así como aquellas entidades que han afectado al OA durante su evolución.	
Versión.	Edición del OA, por ejemplo versión 1.0
Estatus.	Estado de terminación o la condición en la que se encuentra el OA. Bosquejo, final, en revisión, no disponible, etcétera.
Otros colaboradores.	
Rol.	Clase de contribución, debe ser especificado al menos el autor del OA. Autor, director, iniciador, integrador evaluador, redactor, experto, desconocido, etcétera.
Entidad.	Identificación e información de las entidades que contribuyen al OA. Deben ser ordenadas en base a su relevancia. Por ejemplo: doctor Jaime Ortiz, Universidad Autónoma de Aguascalientes.
Fecha.	Fecha de contribución. Por ejemplo: 25/03/06.

Tabla A1.7 Plantilla de la Categoría Educacional.

Categoría Educacional. Agrupa las características educativas y pedagógicas del OA, lo cual es importante para aprovecharlo con calidad.	
Tipo de interactividad.	Modo predominante de aprendizaje, apoyado por el OA. Aprendizaje activo: induce a actividades productivas por parte del usuario. Aprendizaje expositivo: cuando el trabajo del usuario consiste principalmente en comprender los contenidos del exponente por medio de diversos recursos y medios. Aprendizaje mixto. Aprendizaje indefinido.
Tipo de recurso de aprendizaje.	Especifica la clase de OA, considerando la clase de actividad más predominante. Ejercicios, simulación, cuestionarios, diagramas, textos, caso práctico, ejemplos, gráficos, etcétera.
Nivel de interactividad.	El grado de interactividad que caracterice al OA. Muy baja, baja, media, alta, muy alta. “Interactividad” se refiere al grado en el que el usuario puede influir en la aplicación del OA.
Densidad semántica.	La densidad semántica de un OA puede estimarse en función de su tamaño, medida; o de la duración, cuando se trata de archivos multimedia. La densidad semántica de un OA es independiente de su dificultad. Muy baja, baja, media, alta, muy alta.
Fines de uso del usuario.	Usuario principal del OA. Autor, estudiante, administrador, maestro.
Contexto	Nivel educativo, entorno principal dentro del cual se interactuará con el OA. Educación primaria, secundaria, preparatoria, universidad etcétera.
Edad	Edad del usuario promedio.
Dificultad	El nivel de dificultad que implica para los usuarios promedio interactuar con el OA, Muy fácil, fácil, medio, muy difícil, difícil.
Tiempo promedio de aprendizaje	Tiempo aproximado que toman los usuarios para trabajar con el OA.
Descripción	Comentarios de cómo usar el OA.
Lengua	Idioma usado por los usuarios que interactúan con el OA

Tabla A1.8 Plantilla de la Categoría de Derechos de Propiedad.

Categoría derechos de propiedad, Agrupa los derecho intelectuales y las condiciones de uso del OA.	
Costo	Especificar si el uso de este OA requiere de un pago (si/no)
Copyright y otras restricciones	Especificar si se aplica una ley de Copyright o alguna restricción. (si/no)
Descripción	Comentarios sobre las condiciones de uso del OA.

El producto que se obtiene al finalizar esta fase está integrado por el bosquejo del contenido, las plantillas de actividad, evaluación, categoría genera, ciclo de vida, educacional y derechos de propiedad.

A1.1.3 Fase 3 "Desarrollo"

En esta fase se realizarán el armado, empaquetado y almacenamiento del OA, en un repositorio temporal. En esta fase se aborda la parte tecnológica del OA, así que es necesario poner especial atención en explicarla a los usuarios en general que no estén muy familiarizados con la cuestión tecnológica. (Ver Figura A.6)

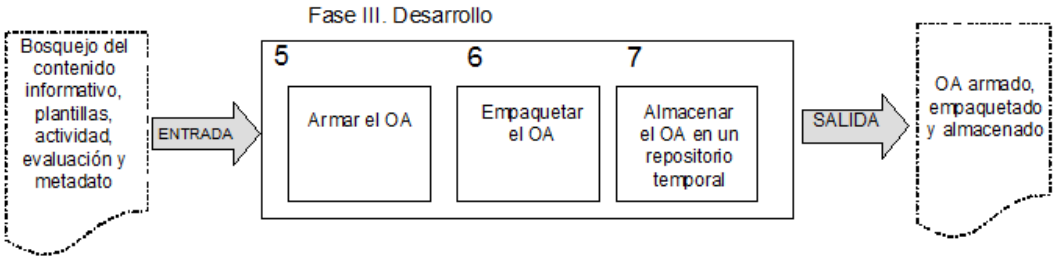


Figura A1.6 Fase 3 "Desarrollo". (Muñoz Arteaga J., Osorio Urrutia B., Álvarez Rodríguez Francisco J., Cardona Salas P., 2006)

Paso 5 "Armar"

Es necesario integrar cada uno de los componentes del OA (identificados en la fase II), en un archivo html, que es recomendable sea una plantilla que puede

contener datos como la información general de la institución que está produciendo los OA, así como el logotipo de la institución. La finalidad de la plantilla es ofrecer a los alumnos contenidos con un formato uniforme. (Ver Figura A.7)

Figura A1.7 Armado del OA. (Figura Modificada de Muñoz Arteaga J., Osorio Urrutia B., Álvarez Rodríguez Francisco J., Cardona Salas P., 2006)

Se puede armar la plantilla con cualquier software generador de HTML o XML, o desde cada uno de estos lenguajes, directamente, es recomendable que se seleccione el lenguaje de su preferencia, o en el cual tenga más habilidades de desarrollo. En caso de no contar con habilidades en programación, habrá que buscar la asesoría de un experto.

Una vez que se tiene el armado del OA, se procederá a empaquetarlo, esto con la finalidad de que quede estandarizado y sea fácil de rehusar en diversas plataformas.

Existen una diversidad de estándares tecnológicos que permiten la manipulación de los OA; esta metodología hace referencia al SCORM por ser uno de los más usados.

Paso 6 "Empaquetar"

El paquete SCORM es un archivo comprimido que contiene a su vez tres archivos del OA, el manifiesto y las hojas de estilo que permiten su interpretación. En este punto, mediante un software generador de paquete SCORM, se procederá a crear y editar el metadato del OA. Existen diversas herramientas de software que puede ayudar en esta actividad, seleccione el que más se adapte a sus necesidades. Para la aplicación de esta metodología

se propone el uso del RELOAD. En este paso se usaran las plantillas general, ciclo de vida, educacional y derechos de autor previamente elaboradas. Al finalizar el proceso de empaquetamiento se generará un archivo zip. En este punto es importante apoyarse de un experto. (Ver Figura A.8)

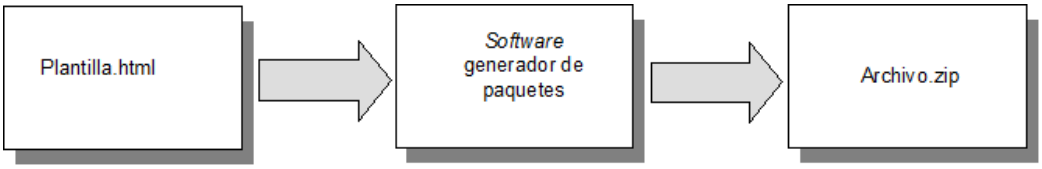


Figura A1.8 Proceso de Empaquetamiento. (Muñoz Arteaga J., Osorio Urrutia B., Álvarez Rodríguez Francisco J., Cardona Salas P., 2006)

Paso 7 "Almacenar el OA en un Repositorio Temporal"

Los repositorios de los OA son aplicaciones que facilitan su almacenamiento, búsqueda, uso y rehúso; para que esto se cumpla, los OA tienen que contar con una estructura adecuada de su metadato. En este paso se subirán los OA generados en un repositorio temporal, de donde serán seleccionados para su evaluación. Dependiendo del repositorio variará el proceso de integración del OA al mismo. El producto final de esta fase es el armado, empaquetado y almacenamiento del OA.

Hasta el momento se ha terminado con el proceso de elaboración del OA, es momento de proceder a su evaluación. Esto se realiza con la finalidad de no tener OA de baja calidad en el repositorio de OA evaluados.

A1.1.3.4 Fase 4 "Evaluación"

En esta fase se procederá a evaluar el OA por un grupo de expertos y almacenarlo en un repositorio de los OA evaluados. (Ver Figura A.9)

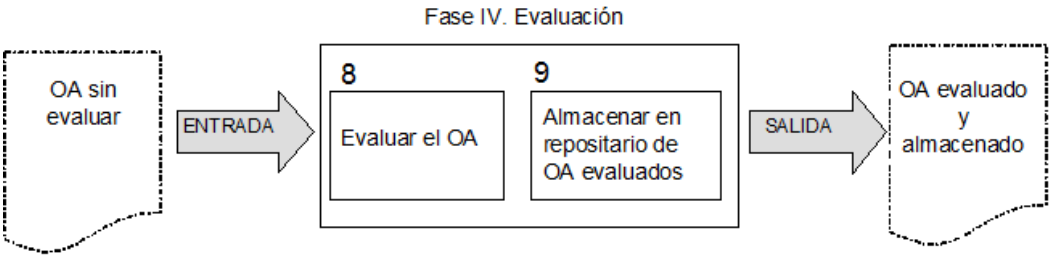


Figura A1.9 Fase 4 "Evaluación". (Muñoz Arteaga J., Osorio Urrutia B., Álvarez Rodríguez Francisco J., Cardona Salas P., 2006)

Paso 8 "Evaluar el OA"

En este paso el OA será evaluado por un grupo de expertos tomando como referencia una serie de indicadores. (Morales E., García Francisco J., Barrón A., Berlanga A. J. y López C., 2005), propone evaluar un OA bajo las siguientes categorías.

- Categoría didáctico-curricular.
- Categoría técnica-estética.
- Categoría funcional.

Estas categorías están íntimamente ligadas con su respectiva categoría del metadato. Se sugiere una elaboración de un instrumento que permita evaluar el OA bajo una serie de indicadores.

Paso 9 "Almacenar el OA en un Repositorio de Objetos de Aprendizaje Evaluados"

Después de la evaluación del OA por un grupo de expertos, éste será almacenado en un repositorio, aquí serán almacenados sólo los OA que cumplan con una mínima calificación determinada por un grupo de expertos.

El producto final de esta fase es el OA evaluado y almacenado. Hasta el momento se garantiza que los OA generados tienen un cierto grado de calidad y que han pasado por un proceso de evaluación, quizás es sumamente arriesgado el término calidad, pero los OA evaluados son mucho más confiables que los primeros porque han pasado por un proceso minucioso de evaluación.

En esta fase se procederá a hacer uso del OA en un contexto determinado. Es deseable que los OA hasta el momento elaborados y almacenados, sean integrados en cursos que estén disponibles en un sistema de gestión de aprendizaje. (Ver Figura A.10)

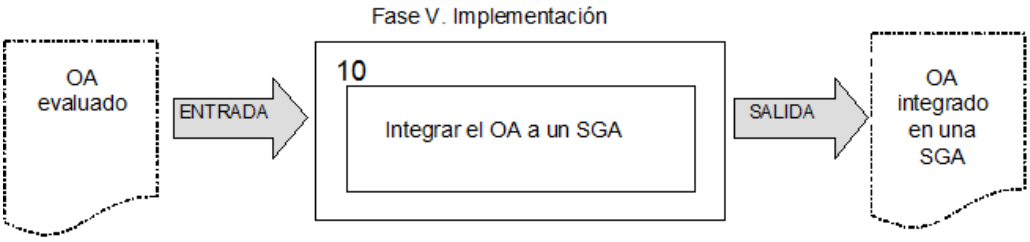


Figura A1.10 Fase 5 "Implementación".

Paso 10 Integrar el OA a un Sistema de Gestión de Aprendizaje (SGA)

Al integrar el OA a un SGA, se añadirá a un determinado contexto, donde se podrá realizar una explotación adecuada del mismo. Para efectos de la presente metodología, los OA generados serán integrados al SGA (Moodle). Se optó por este sistema (Moodle) además de ser software libre, en un número importante de instituciones de educativas de nivel nacional e internacional está siendo aplicado, por ser fácil de aprender y usar , como ejemplo tenemos a la Universidad Autónoma de Aguascalientes (UAA) en la actualidad ofrece cursos de educación a distancia a través de dicha plataforma. Una vez que los OA estén integrados al contexto determinado podrán ser evaluados por los consumidores, la retroalimentación que hagan ambos del mismo será muy valiosa, ya que esto da una serie de indicadores para obtener una mejora continua de los mismos.

El producto final que se obtiene de esta fase es el OA integrado en un SGA, el cual será constantemente evaluado.

A1.2 Repositorios de Objetos de Aprendizaje (ROA)

Los OA actualmente tienen como sistemas formales de acceso dos instancias, los contenedores de OA, conocidos como “repositorios digitales” y los sistemas administradores de cursos, CMS (Course Management System) también conocidos como LMS (Learning Management System). En ambos casos es necesario un estándar de intercambio de información.

Es por lo anterior que existen esfuerzos a nivel mundial se están llevando para producir una metodología estandarizada (se encuentran en la etapa de proyectos de investigación) para la construcción de ROA, es de resaltar que gran número de las iniciativas son para formar redes con estos repositorios. El interés general de la comunidad está siendo enfocado a conectar y utilizar recursos distribuidos en repositorios heterogéneos (Hatala M., Richards G., Eap. T. & Willms J., 2004), se está trabajando en corregir las divergencias de los repositorios para canalizarse en el uso de los mismos estándares o al menos el uso de los que sean compatibles con otros. También se está trabajando en la creación de repositorios federados (Massart D. & Dung L. T., 2004), que se basan en búsquedas propagadas en metadatos distribuidos en distintos servidores. El objetivo final es la interoperabilidad, es decir, que los sistemas tengan la capacidad para trabajar fácilmente con otro u otros. Entre estos sistemas se busca, a través de una normalización tecnológica o procedimental, poder interconectarse para realizar diversas transacciones o actividades, siendo la principal actividad el intercambio de contenidos.

El grupo de especificaciones más ambicioso en e-learning y que está teniendo trabajos específicos para la estandarización de los repositorios es IMS (Instructional Management Systems), que dentro de sus propuestas maneja IMS Digital Repositories Specification y otras especificaciones que se relacionan con ésta, haciendo posibles, con su implementación, la interoperabilidad de los repositorios en los ambientes e-learning y fuera de éstos.

Como se menciono en el capítulo 3, para este trabajo de tesis adoptaremos el repositorio propuesto por el grupo de investigación de Objetos de Aprendizaje de la Universidad Autónoma de Aguascalientes LORAA (Learning Object Repository of Universidad Autónoma de Aguascalientes). Para facilitar el intercambio de contenidos existentes en su plataforma y además porque ya existe una vinculación en este tema con otras instituciones educativas como la Universidad de Veracruzana, la Universidad de Colima, entre otras.

Es importante mencionar que en el diseño de los Objetos de Aprendizaje se tiene que considerar:

1) Tomar en cuenta las especificaciones IMS relacionadas con la estandarización de los ROA son: IMS Digital Repositories Specification, IMS Content Packaging, IMS Learning Resource Metadata e IMS Resource List Interoperability.

2) NO olvidar de incluir en esta información en la lista de recursos de un OA a IMS RLI para abrir el vínculo de comunicación entre las bibliotecas automatizadas y los ROA.

Todo lo anterior para facilitar la interoperabilidad y así mismo facilitar la exportación e importación de contenidos.

El IMS CP de SCORM hace compatible a los paquetes con otras aplicaciones desarrolladas sobre IMS de otros Repositorios, ambas descripciones funcionan con un derivado de LOM para la definición de su esquema de metadatos para permitir el intercambio, el almacenamiento de recursos y su conexión con otras aplicaciones.

Anexo 2 “Manual de la Interfaz para el Proyecto ProgEst”

Introducción

Este manual se desarrolló de acuerdo a la propuesta de Velasco-Santos, Laureano-Cruces, Mora-Torres y Sánchez-Guerrero, (2009), como parte del proyecto ProgEst para explicar el diseño de la interfaz del ProgEst la cual permite la interacción del estudiante con los contenidos, para llevar a cabo esto es necesario contar con una interfaz atractiva y funcional. (Lamarca, 2006)

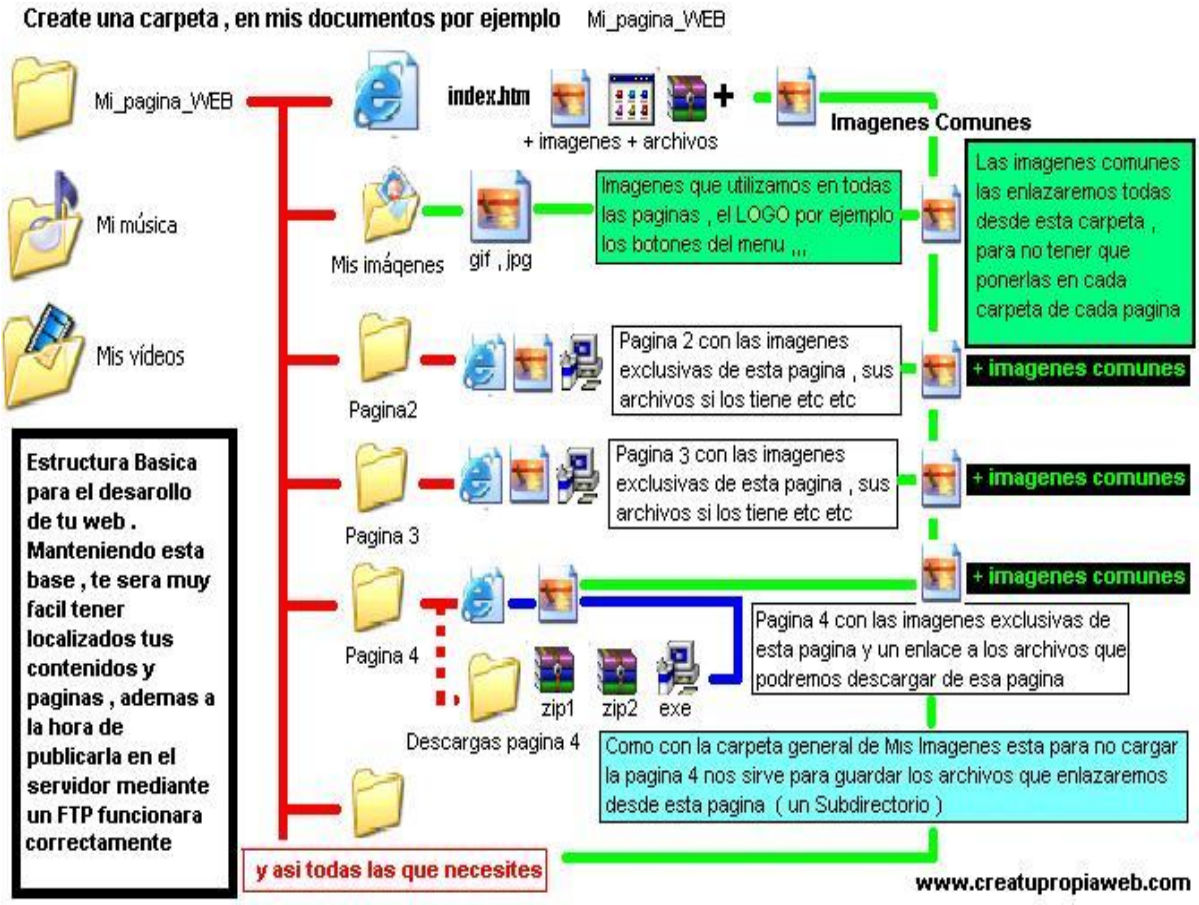
De acuerdo a lo anterior, son dos los encargados de crear los diseños de interfaces eficaces: los diseñadores gráficos y los desarrolladores deben tomar en cuenta el diseño visual, la interacción y la arquitectura del contenido. (Krug, 2006)

Este manual contiene el diseño y la definición los lineamientos de uso de las ***Pantallas de Presentación del ProgEst***, su aplicación para cada caso del proyecto como retícula para el o los cuestionarios, presentación de la pantalla del curso en este último caso aplicando la psicología del color (Sutton y Whelan Bride, 2006) y estilo de aprendizaje del estudiante (Activo, Reflexivo, Teórico y Pragmático). Así mismo en este documento se describen los lineamientos para la construcción y adecuación de estas pantallas de presentación.

Objetivo

Con este manual tiene como propósito apoyar a mantener actualizada la información contenida en la interfaz del Proyecto “*ProgEst*” y garantizar su vigencia y óptimo funcionamiento, así como informar y guiar a las personas que estén involucradas en darle continuidad al proyecto.

A2.1 Mapa del Sitio



A2.2 Colores para la Interfaz

Colores para la Interfaz del Login

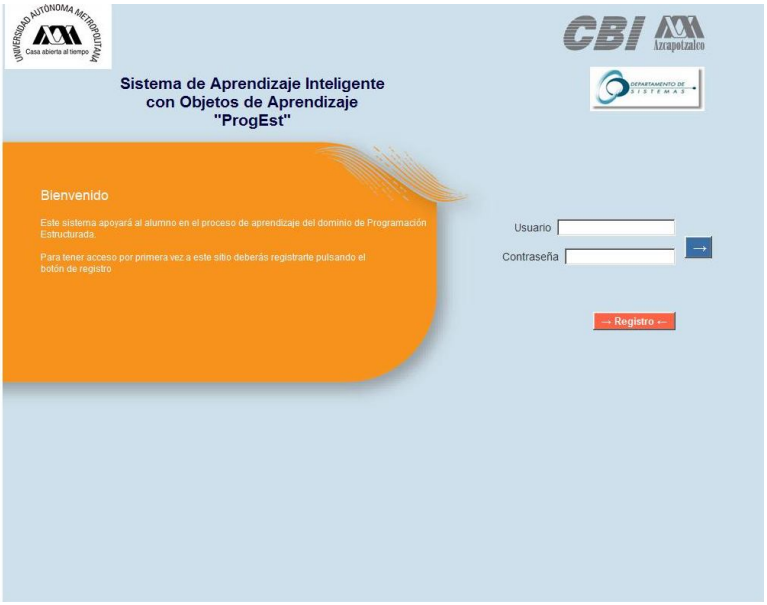


Figura A2.1 Pantalla de la Interfaz del Login del ProgEst.

Como “primera pantalla” y entrada al sistema requiere de un alto grado de impacto, y además de lo anterior debe comunicar el área de estudio en que se enfoca (en este caso alumnos de las carreras de ingeniería) cuyos colores en su mayoría son azul y gris y están estrechamente ligados con conceptos de innovación y tecnología. (Ver Figura A2.1)

De acuerdo a Eiseman, (2006), el color Gris (metálico) es asociado a la sabiduría, madurez, opulencia; es asociado con ideas de avance científico y tecnológico, ideas de vitalidad y competencia. Con base a lo anterior se determinó la siguiente distribución de colores: (ver Tabla A2.1)

Tabla A2.1 Colores de la Interfaz del Login del Sistema ProgEst.

Descripción	Colores
Fondo	Gris 98 (c11, m0, y0, k8)

Botones con Flecha	Azul 66 (c100, m60, y0, k25)
Botón de Registro	Rojo (c0, m92, y100, k10)
Cuadro de Bienvenida	Anaranjado (c0, m51, y100, k0)

Esta combinación proporciona una atmósfera de profesionalismo y seriedad adecuada para los visitantes, ya que está enfocado al sector universitario cuyas edades oscilan en promedio entre los 18 y 24 años. (Eiseman, 2006)

Colores para la Interfaz de los Cuestionarios

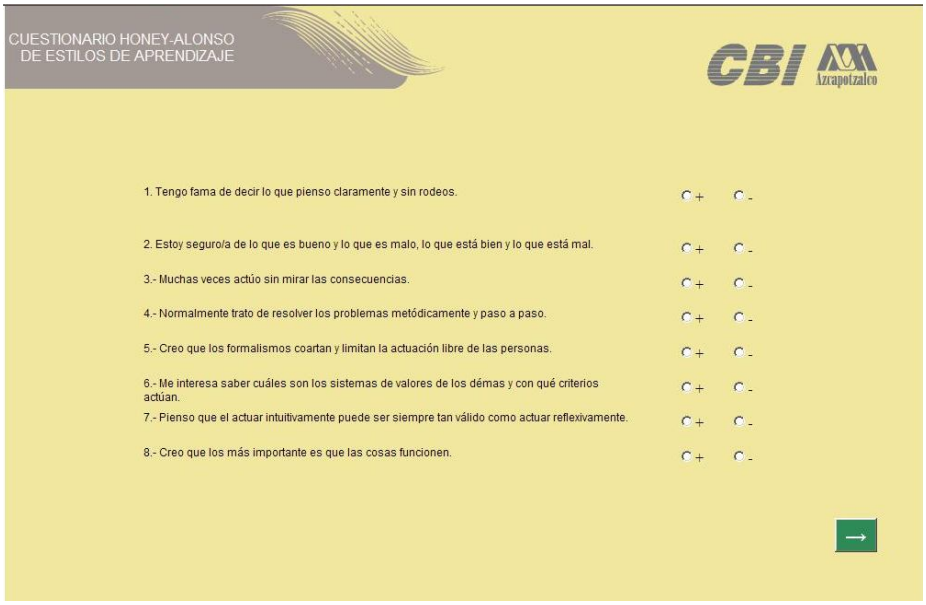


Figura A2.2 Pantalla de la Interfaz de los Cuestionarios de Estilos de Aprendizaje y Orientación de Motivación del Estudiante.

Los cuestionarios utilizados en este Sistema son extensos, por ello se necesitan colores acogedores para formar una atmósfera de confianza para el usuario (con una combinación neutra) y con movimiento para crear una sensación de agilidad y avance. Por lo cual se determinó la siguiente distribución de colores, (ver Tabla A2.2 y Figura A2.2).

Tabla A2.2 Colores de la Interfaz de los Cuestionarios de Estilos de Aprendizaje y de Motivación de la Orientación del Estudiante.

Descripción	Colores	Estado de Animo
Fondo	Gris pantone warm gray 6 (c11, m16, y18, k32)	Armonía
Copete	Amarillo pantone 460 (c2, m2, y43, k3)	Optimismo, Curiosidad
Acentos Botones	Café pantone 465 (c7, m27, y55, k22)	

Colores para la Interfaz de cada Estilo de Aprendizaje

Los colores para cada una de las interfaces de acuerdo al Estilo de Aprendizaje se pueden ver en las Tablas (A2.3, A2.4, A2.5 y A2.6).

Tabla A2.3 Colores de la Interfaz del Estilo de Aprendizaje *"Activo".*

Estilo	Inteligencia	Explicación	Características	Colores	Combinaciones
Activo	Musical Cinético-corporal Espacial Interpersonal Naturista	Se implican plenamente sin prejuicios en nuevas experiencias Son de mente abierta Nada escépticos Piensan que por lo menos una vez hay que intentarlo todo Se crecen ante los desafíos de nuevas experiencias Se aburren con los largos plazos Personas sociales Se involucran Entran en las actividades de los demás	Animador Improvisador Descubridor Arriesgado Espontáneo Creativo Novedoso Aventurero Renovador Inventor Vital Vividor de la experiencia Generador de ideas Lanzado Protagonista Chocante Innovador Conversador Líder Voluntario Divertido Participativo Competitivo Deseoso de aprender Solucionador de problemas Cambiante	Amarillo Rojo Naranja	Pág. 109 (2,2) Fondo pant 130 Copete pant 151 Botones pant 180 Pág. 109 (3,4) Fondo pant 1788 Copete pant 1235 Botones pant 605 Pág. 69 (3,2) Fondo pant 704 Copete pant 1525 Botones pant 458 Pág. 91 (1,1) Fondo pant orange 021 Copete pant process magenta Botones pant process yellow Pág. 91 (1,4) Fondo pant 108 Copete pant purple Botones pant 1785

Tabla A2.4 Colores de la Interfaz del Estilo de Aprendizaje *"Reflexivo".*

Estilo	Inteligencia	Explicación	Características	Colores	Combinaciones
Reflexivos	Musical Lingüística Lógico-matemática Espacial Intrapersonal Pictórica Espiritual	Observar y analizar las experiencias desde diferentes perspectivas Recogen datos analizándolos con detenimiento antes de llegar a alguna conclusión Son prudentes Muy analíticos Disfrutan observando la actuación de los demás y no intervienen hasta que se adueñan de la situación Crean a su alrededor un aire ligeramente distante y condescendiente	Ponderado Concienzudo Receptivo Analítico Exhaustivo Observador Recopilador Paciente Cuidadoso Detallista Elaborador de argumentos Previsor de alternativas Estudioso de comportamientos Registrador de datos Investigador Asimilador Escritor de informes y/o declaraciones Lento Distante Prudente Inquisidor Sondeador	Rosa Marrón Violeta Verde	Pág. 101 (6,4) Fondo pant 204 Copete pant 2567 Botones pant 577 Pág. 73 (6,1) Fondo pant 5145 Copete pant 692 Botones pant 557 Pág. 105 (2,1) Fondo pant 480 Copete pant 479 Botones pant 616 Pág. 77 (3,1) Fondo pant 257 Copete pant cool gray 4 Botones pant 535 Pág. 67 (2,3) Fondo pant 537 Copete pant 441 Botones pant 4545 Pág. 71 (2,1) Fondo pant 468 Copete pant 501 Botones pant 452

Tabla A2.5 Colores de la Interfaz del Estilo de Aprendizaje *"Teórico".*

Estilo	Inteligencia	Explicación	Características	Colores	Combinaciones
Teóricos	Lingüística Lógico-matemática Espacial Intrapersonal	Aceptan e integran las observaciones dentro de teorías lógicas y complejas Enfocan los problemas en vertical escalonada en etapas lógicas Tienden a ser perfeccionistas Integran los hechos en teorías coherentes Les gusta analizar y sintetizar Son profundos en su sistema de pensamiento para establecer principios, teorías y modelos Si es lógico es bueno Buscan la racionalidad y objetividad huyendo de lo subjetivo y ambiguo	Metódico Lógico Objetivo Crítico Estructurado Disciplinado Planificado Sistemático Ordenado Sintético Razonador Pensador Relacionador Perfeccionista Generalizador Buscador de hipótesis De teorías De modelos De preguntas De supuestos subyacentes De conceptos De finalidad clara De racionalidad De "por que" De sistemas de valores, de criterios,… Inventor de procedimientos para… Explorador	Azul Gris	Pág. 67 (1,2) Fondo pant 5635 Copete pant 5405 Botones pant 646 Pág. 73 (3,2) Fondo pant 5415 Copete pant 442 Botones pant 422 Pág. 77 (3,4) Fondo pant 431 Copete pant 289 Botones pant 429 Pág. 97 (4,3) Fondo pant 5425 Copete pant 5445 Botones pant 5395 Pág. 97 (6,1) Fondo pant 9060 Copete pant 429 Botones pant 432

Tabla A2.6 Colores de la Interfaz del Estilo de Aprendizaje *"Pragmático".*

Estilo	Inteligencia	Explicación	Características	Colores	Combinaciones
Pragmáticos	Musical Cinético-corporal Espacial Naturista	Aplicación práctica de ideas Descubren el aspecto positivo de las nuevas ideas y aprovechan la primera oportunidad para experimentarlas Les gusta actuar rápidamente y con seguridad con aquellas ideas y proyectos que les atraen Tienden a ser impacientes	Experimentador Práctico Directo Eficaz Realista Técnico Útil Rápido Decidido Planificador Positivo Concreto Objetivo Claro Seguro de sí Organizador Actual Solucionador de problemas Aplicador de lo aprendido Planificador de acciones	Marrón Rojo Amarillo Verde	Pág.111 (1,2) Fondo pant 234 Copete pant 371 Botones pant 732 Pág. 69 (1,1) Fondo pant 111 Copete pant 497 Botones pant 1395 Pág. 87 (3,2) Fondo pant 1807 Copete pant 469 Botones pant 3995 Pág. 109 (1,4) Fondo pant 193 Copete pant 612 Botones pant 1235 Pág. 109 (2,1) Fondo pant 131 Copete pant warm red Botones pant 612

A2.3 Criterios Uso para la Interfaz

Normas de Uso de la Pantalla de la Interfaz

Las modificaciones a la página de la interfaz deberán seguir las siguientes reglas para el script de la página:

- Utilizar minúsculas.
- No espacios.
- No acentos.
- El separador es “_”.
- Cambiar “ñ” por “ni”.
- Nunca borrar archivos html.

Identificadores

1. En negritas capa de primer nivel.
2. En cursiva capa de segundo nivel.
3. Subrayado capa de tercer nivel.
4. Entre corchetes donde se encuentra el archivo dentro de la página.
5. Entre llaves se encuentran comentarios sobre el archivo ya sea para identificarlo o la forma de administrar dicho archivo ya que hay archivos que no se van a desechar, sino que se van a guardar.
6. Lo que tenga guión al principio son los botones o los vínculos dentro de la página, es decir donde se encuentra cada archivo.
7. Los archivos con # son vínculos a la misma página.

Tipos de Archivos

Sólo se utilizarán archivos con extensiones:

- .giff archivo de imagen.
- .jpg archivo de imagen.
- .png archivo de imagen.
- .pdf archivo de Adobe Acrobat.
- .ppt archivo de Power Point.
- .swf archivo de Flash.
- .html archivo para estructurar textos.
- .wav archivo de audio.

En la pantalla de Login se ubican 5 imágenes:

1. El fondo del recuadro de bienvenida.
2. Estado en reposo del botón de ingreso.
3. Estado sobre del botón de ingreso.
4. Estado en reposo del botón de registro.
5. Estado sobre del botón de registro.

En las pantallas de cuestionario se ubican 6 imágenes:

1. El copete de la pleca superior.
2. La de logos en la esquina superior derecha.
3. Estado en reposo del botón de avance.
4. Estado sobre del botón de avance.
5. En el cambio de cuestionarios:
    - Estado en reposo del botón de test 2.
    - Estado sobre del botón de test 2.

En las pantallas de contenido o “estilos de aprendizaje” se ubican 10 imágenes:

1. El copete del menú.
2. La de logos.
3. La pleca a la izquierda de la frase.
4. La pleca a la derecha de la frase.
5. La pleca a la izquierda de la numeración del submenú.
6. La pleca a la derecha de la numeración del submenú.
7. Estado en reposo del botón de inicio del submenú.
8. Estado sobre del botón de inicio del submenú.
9. Estado en reposo del botón de ayuda del submenú.
10. Estado sobre del botón de ayuda del submenú.

A2.4 Retícula

Para este proyecto se usan 3 tipos de retícula diferentes, para el ***Login***, el ***cuestionario*** y propiamente ***la interfaz del contenido del curso***.

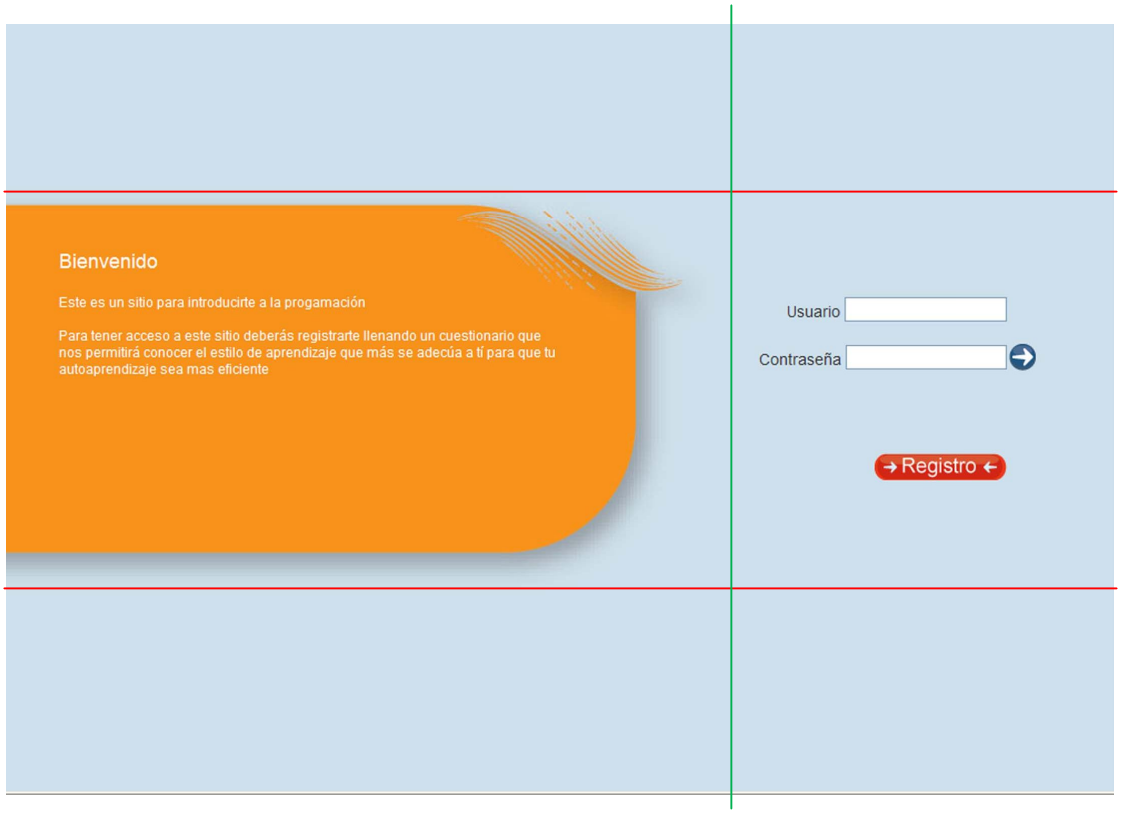
La medida de la página es de 1000px de ancho por 650px de largo, por lo que todas las retículas conservarán esta medida en su tabla principal.

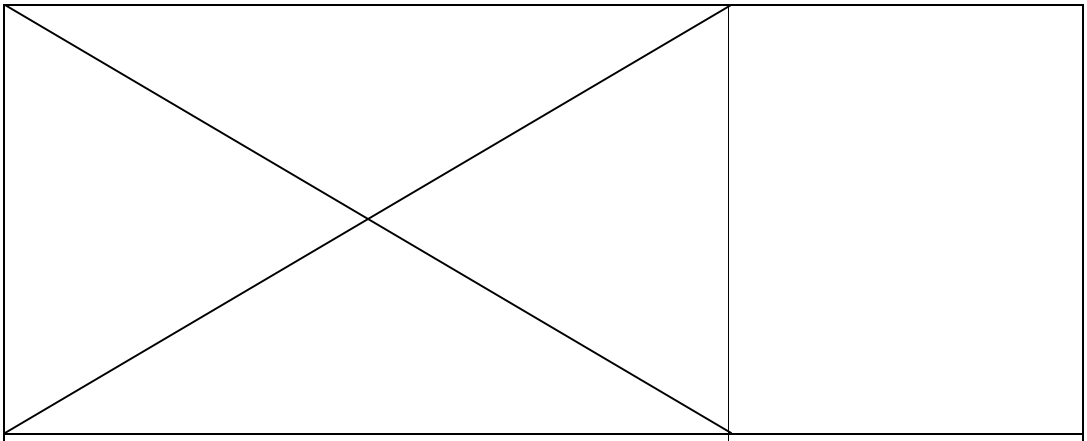
Retícula del Login

- La tabla principal consta de 2 columnas y 3 filas.
- La primera columna mide 655px de ancho.
- La segunda columna mide 345px de ancho.
- La primera fila mide 150px de altura.
- La segunda fila mide 364px de altura.

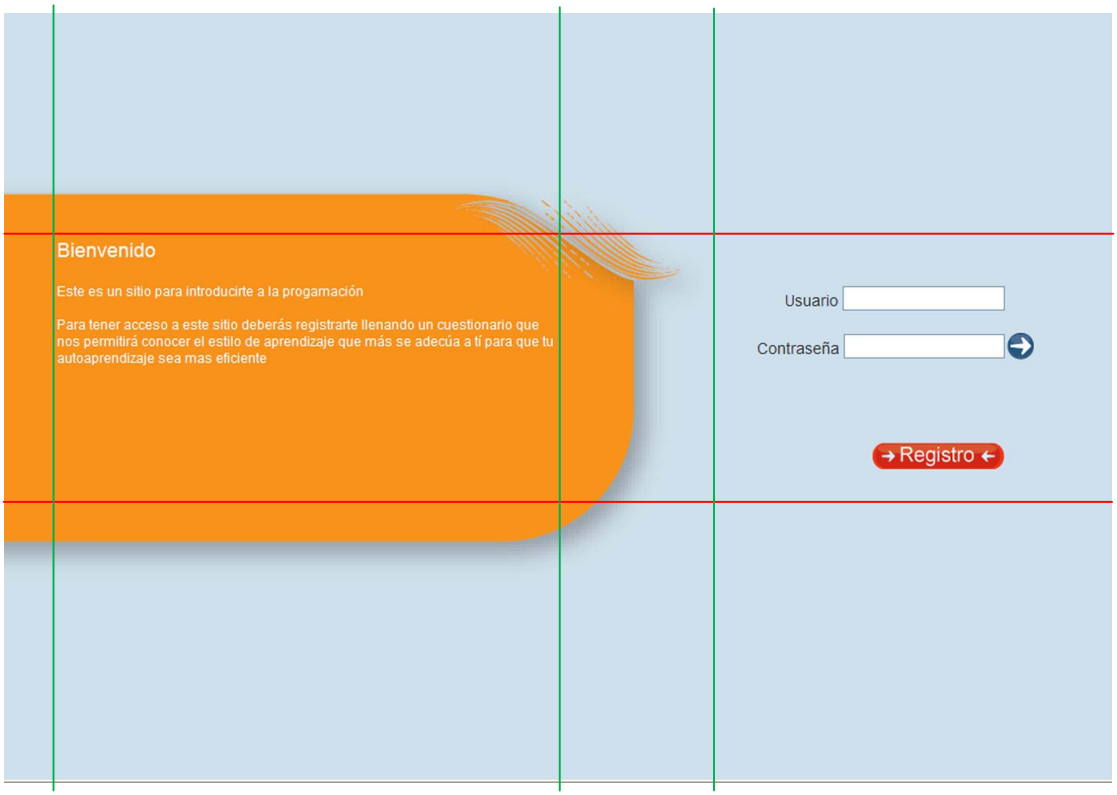
- La tercera fila mide 254 px de altura.

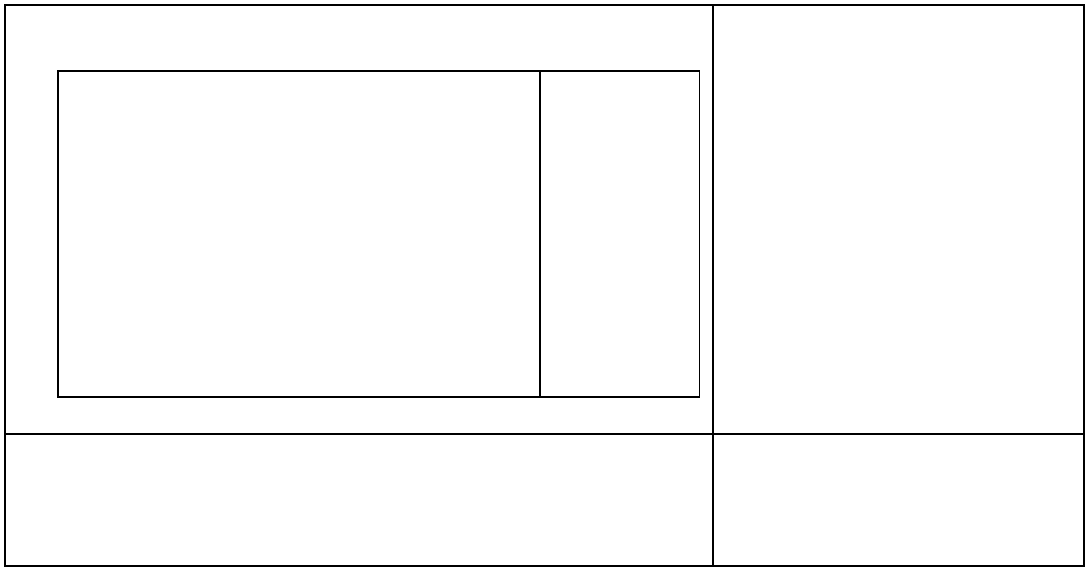
- Se unen las columnas de la primera fila y las de la tercera
- En la primera columna, fila 2, está la imagen “loginJPG.jpg” (655px de ancho y 364px de altura). (Ver Figura A2.1)



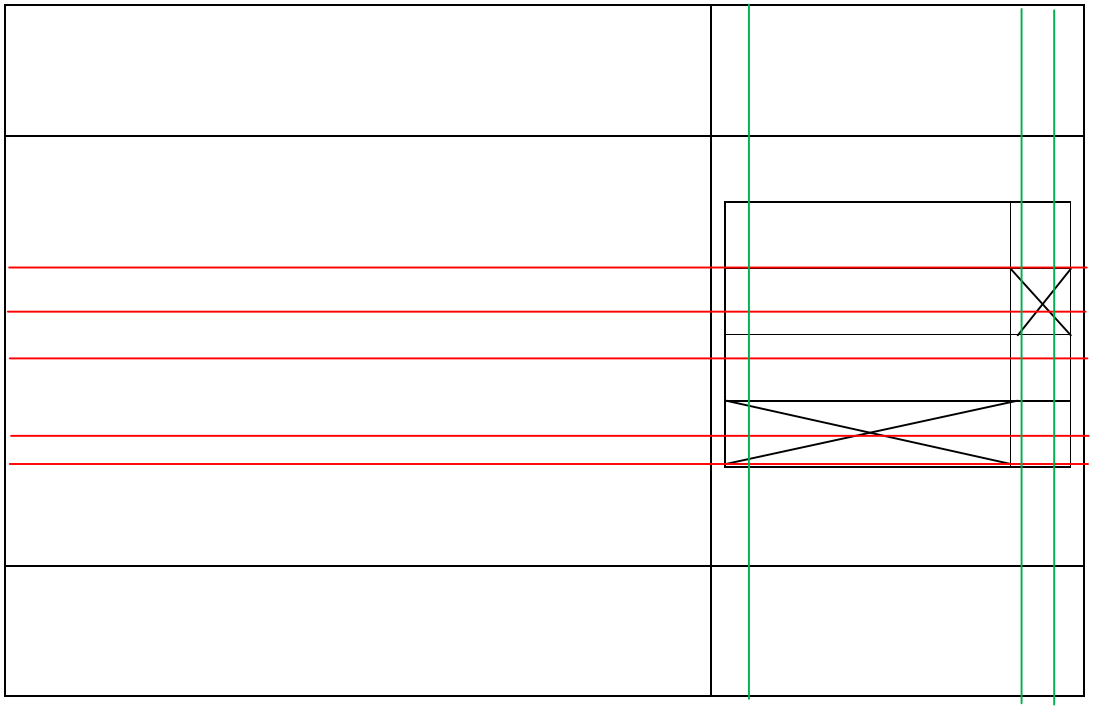


En esta misma celda está insertada una tabla que contendrá el texto de bienvenida, ésta mide 500px de ancho por 250px de altura y tiene 2 columnas La primera mide 50px de ancho y la segunda mide 450px de ancho, donde se escribe el texto y la segunda queda vacía.





En la segunda fila y segunda columna, hay otra tabla con justificación a la izquierda que tiene 2 columnas y 4 filas; mide 274px de ancho y 170px de largo.



- La primera columna mide 226px de ancho y la segunda mide 24px de ancho.
- La primera fila mide 35px de altura.

- La segunda fila mide 35px de altura.
- La tercera fila mide 55px de altura.
- La cuarta fila mide 25px de altura.

En la primera columna, fila 4 de esta tabla se encuentran las imágenes en roll over “LoginBTN-registro-INACT.png” y “LoginBTN-registro-ACT.png” (119px de ancho y 24px de altura), con justificación de celda a la derecha y al fondo.

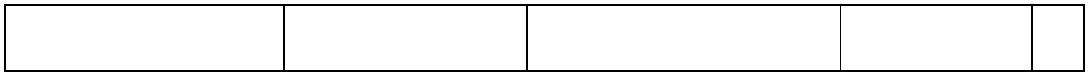
En la segunda columna, fila 2 están las imágenes en roll over “LoginBTN-registro-INACT.png” y “LoginBTN-registro-ACT.png” (24px de ancho y 24 px de altura), con justificación de celda centrada y arriba.

Retícula de los Cuestionarios

Los cuestionarios están en una retícula de 5 columnas y 5 filas. (Ver Figura A2.3)

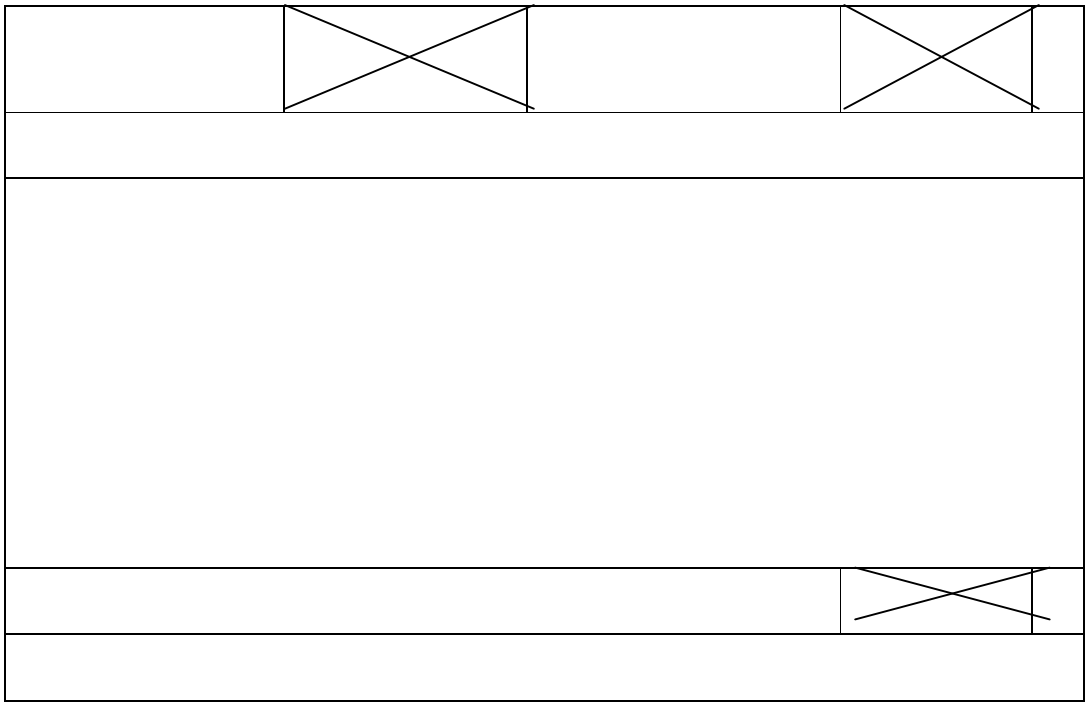
Figura A2.3 Pantalla de la Interfaz de Registro de los Datos del Estudiante.

- La primera columna mide 250px de ancho.
- La segunda columna mide 230px de ancho.
- La tercera columna mide 285px de ancho.
- La cuarta columna mide 185px de ancho.
- La quinta columna mide 50px de ancho.
- La primera fila mide 102px de altura.
- La segunda fila mide 50px de altura.
- La tercera fila mide 448px de altura.
- La cuarta fila mide 50px de altura.
- La quinta fila mide 50px de altura.



Se unen todas las columnas de la segunda, tercera y quinta filas.

En la cuarta fila se unen las tres primeras columnas de izquierda a derecha.



En la fila 1, columna 2, está la imagen “C-cuest1.gif” (230px de ancho y 90px de altura) y en la columna 4 está la imagen “logo-gris.png” (185px de ancho y 49px de altura), con justificación de celda centrada y al fondo.

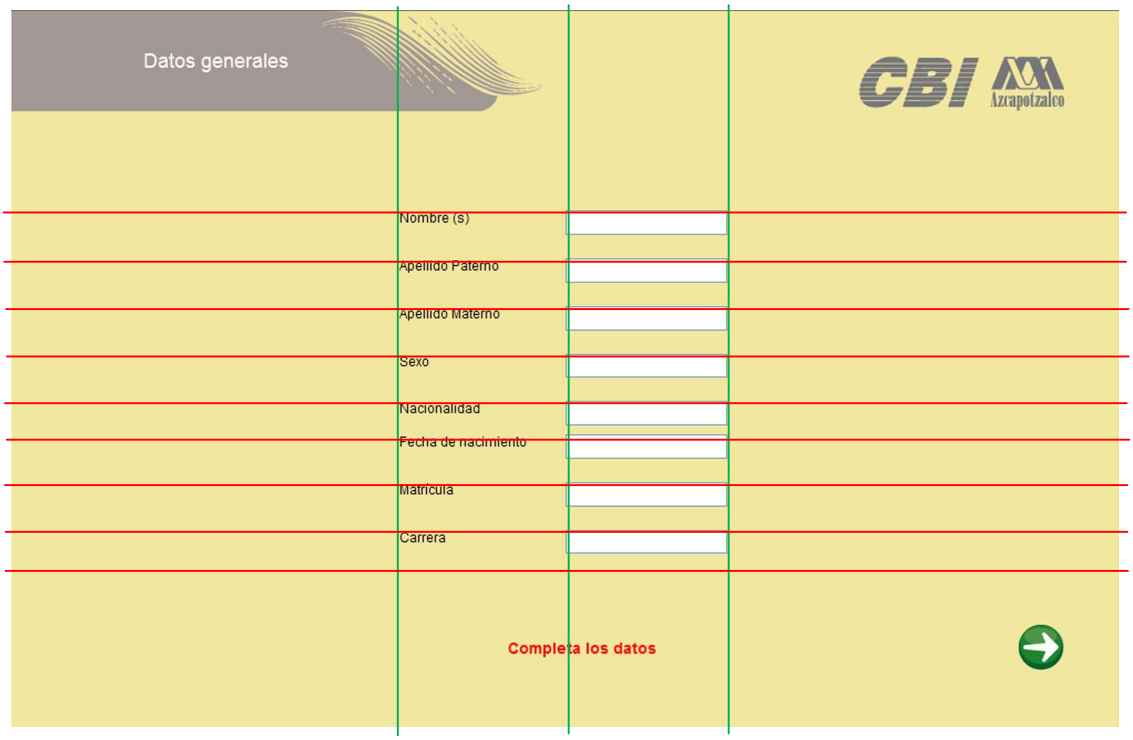
En la fila 4 (donde se unieron las 3 columnas) se coloca una tabla alineada a la derecha (en la cuarta fila) de 1fila y 1 columna, en ésta celda aparecerá el filtro de seguridad con un mensaje de “ERROR” cuando no se responda alguna pregunta.

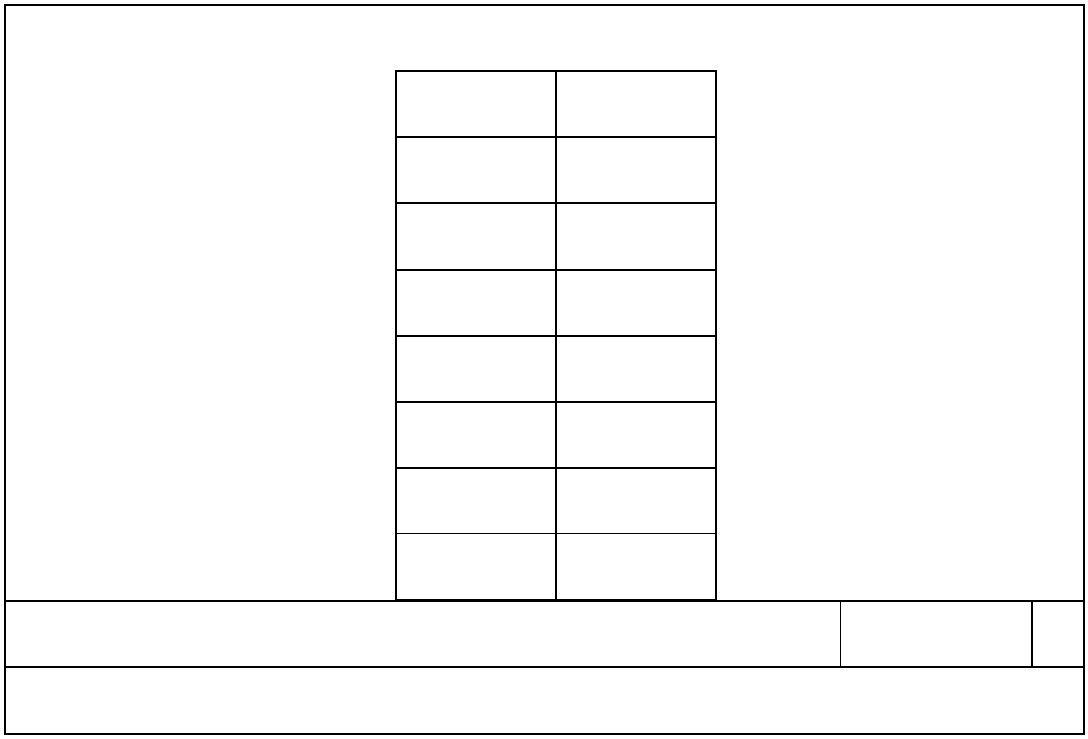
En la fila 4, columna 2, están en roll over las imágenes “CuestBtn-INACT.png” y “CuestBtn-ACT.png” (25px de ancho y 25px de altura), con justificación ala derecha y en medio.

La interfaz permanece durante todo el cuestionario, pero el contenido cambia, por lo que se introducen tablas dentro de la tercera celda, la de mayor tamaño.

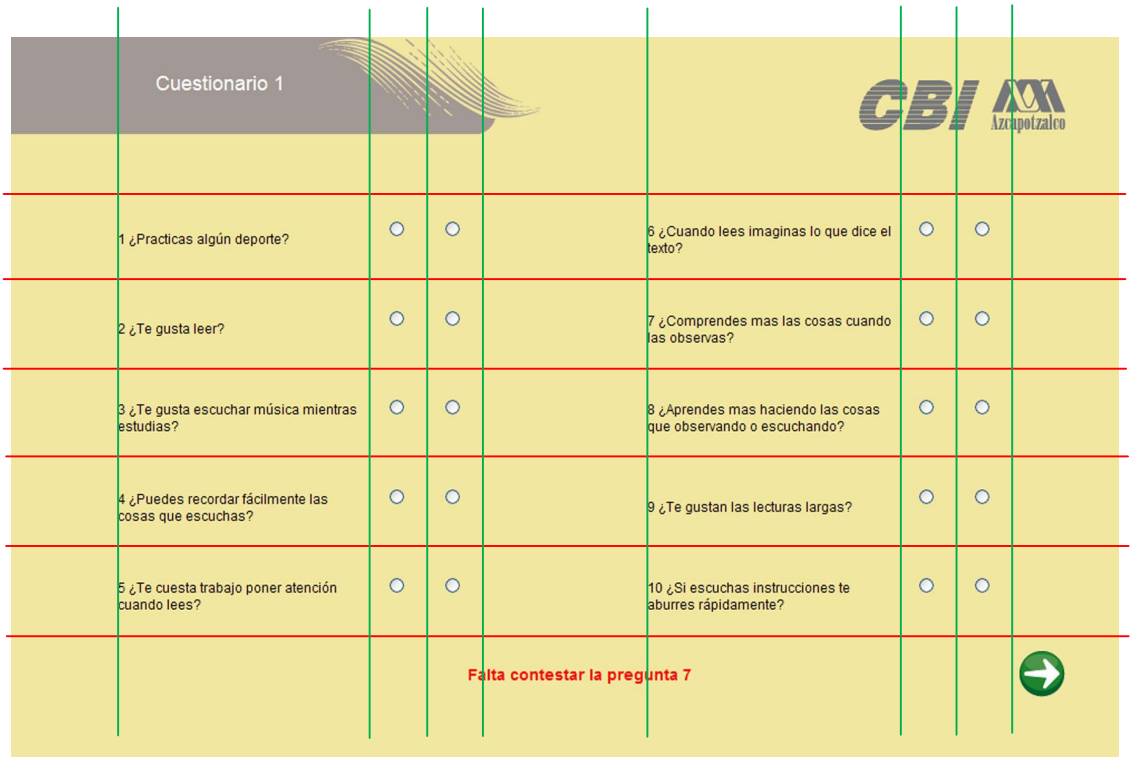
Por lo cual en la primera pantalla del cuestionario aparecen los datos personales y se colocan en una tabla con justificación centrada que mide 300px de ancho por 320px de largo, ésta tabla consta de 2 columnas y 8 filas:

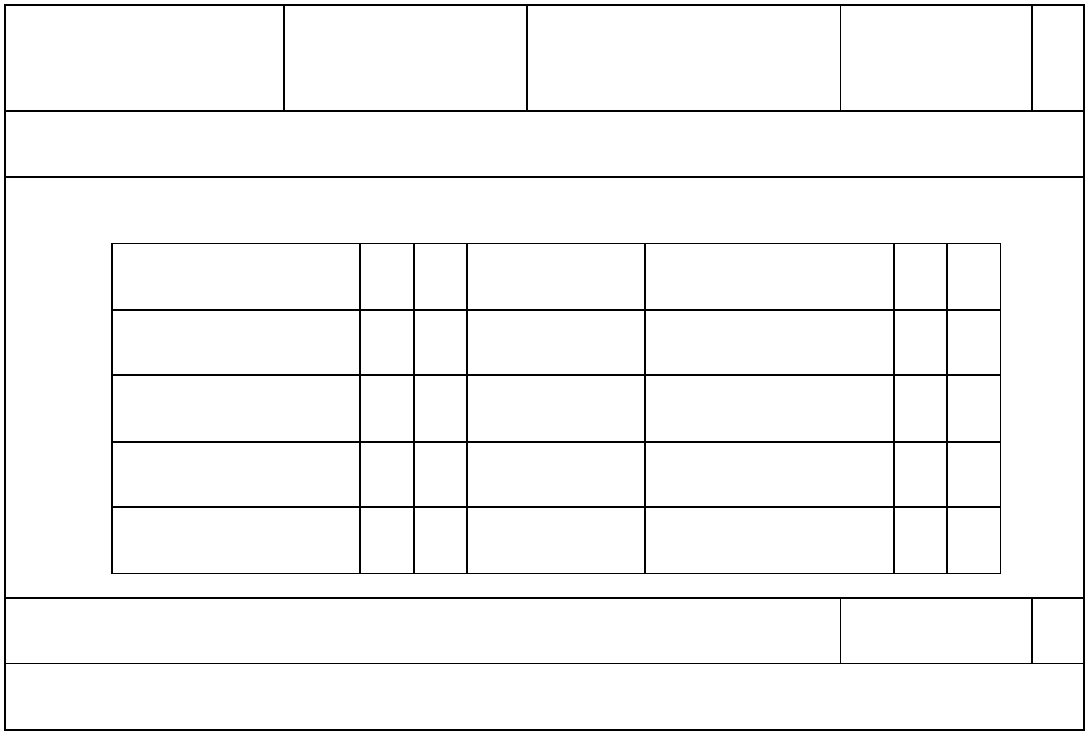
- La primera columna mide 150px de ancho.
- La segunda columna mide 150px de ancho.
- Y todas las filas miden 30px de altura.





En el primer cuestionario se coloca una tabla centrada de 7 columnas y 5 filas de 800px de ancho por 400px de altura.





- La primera columna mide 225px de ancho.
- La segunda columna mide 50px de ancho.
- La tercera columna mide 50px de ancho.
- La cuarta columna mide 150px de ancho.
- La quinta columna mide 225px de ancho.
- La sexta columna mide 50px de ancho.
- La séptima columna mide 50px de ancho.
- Todas las filas miden 80px de altura.

En la cuarta fila (donde se unieron las 3 columnas) se coloca una tabla alineada a la derecha (en la cuarta fila) de fila y 1 columna, en ésta celda aparecerá el filtro de seguridad con un mensaje de “ERROR” cuando no se responda alguna pregunta.

En el cambio de cuestionarios no se agregan tablas, sólo se escribe en la tercera fila el mensaje con justificación centrada. (Ver Figura A2.4)

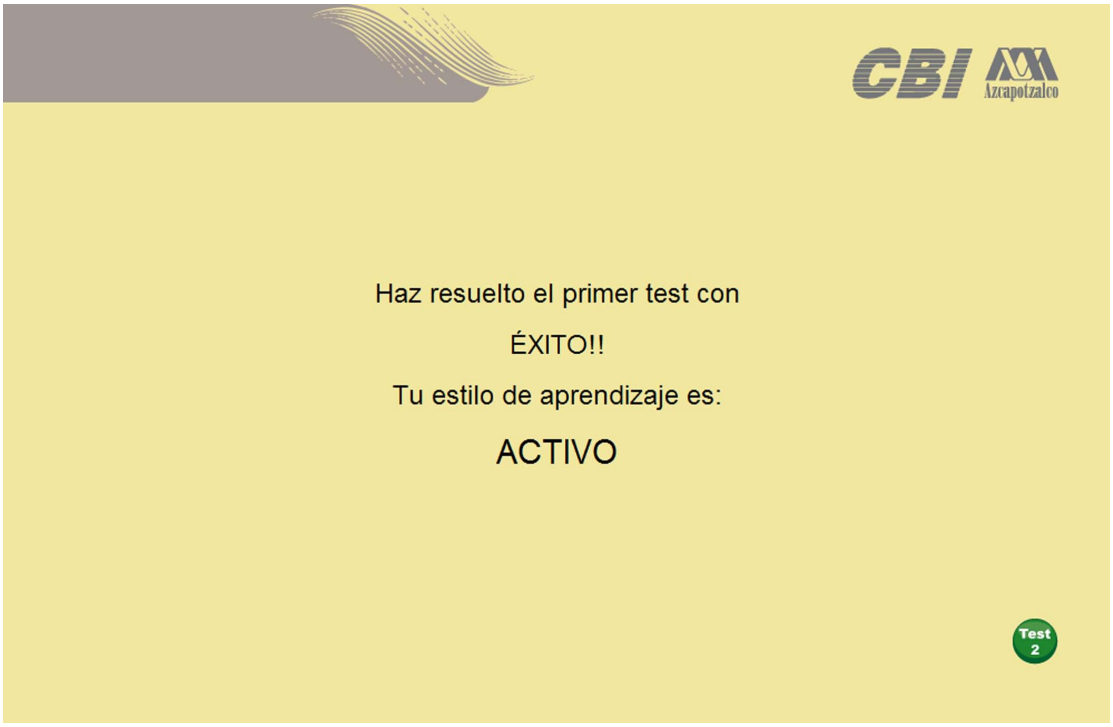
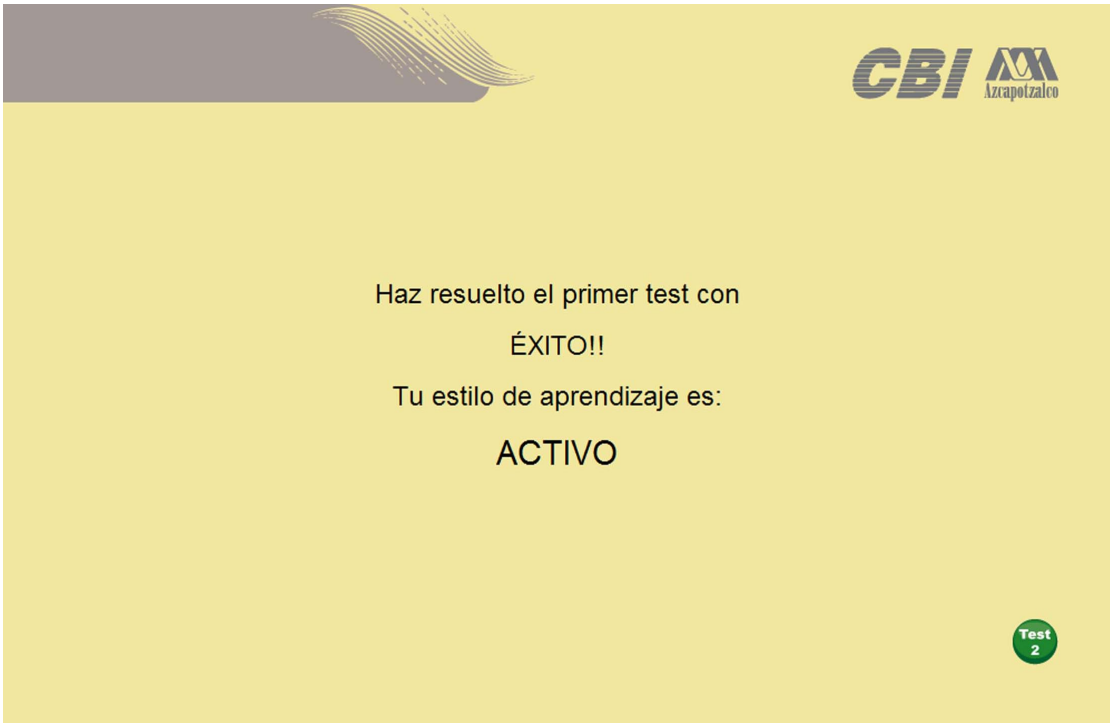
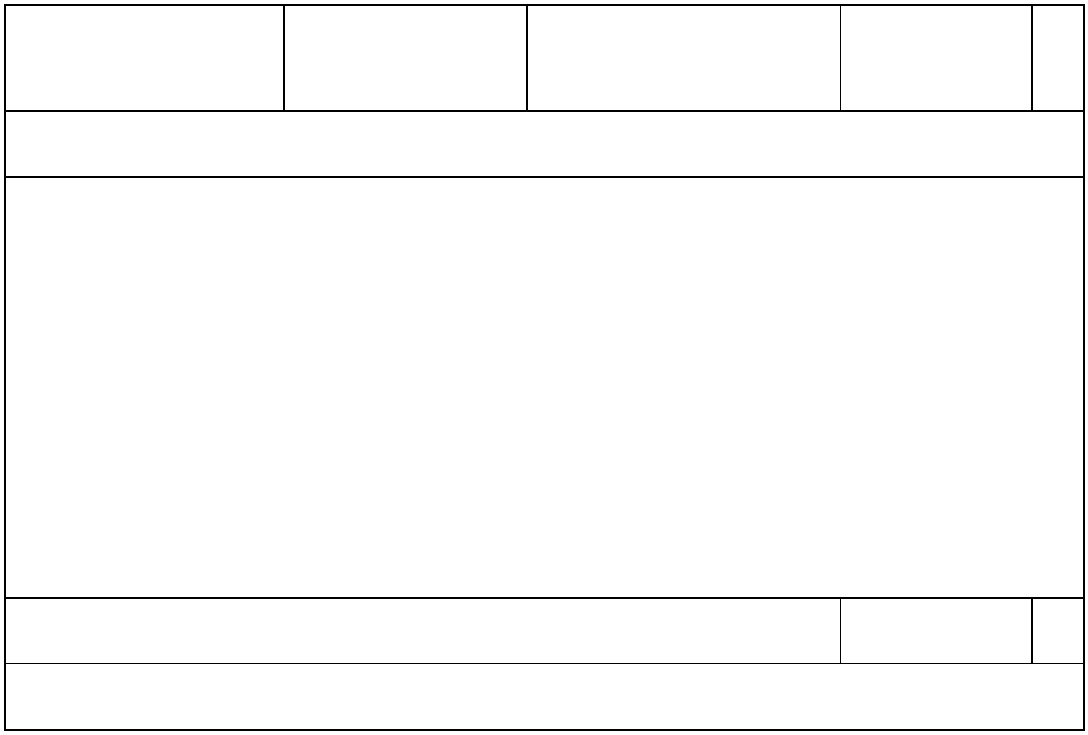


Figura A2.4 Pantalla que muestra el Estilo de Aprendizaje.





El cuestionario 2 tiene la misma estructura del cuestionario 1. (Ver Figura A2.5)

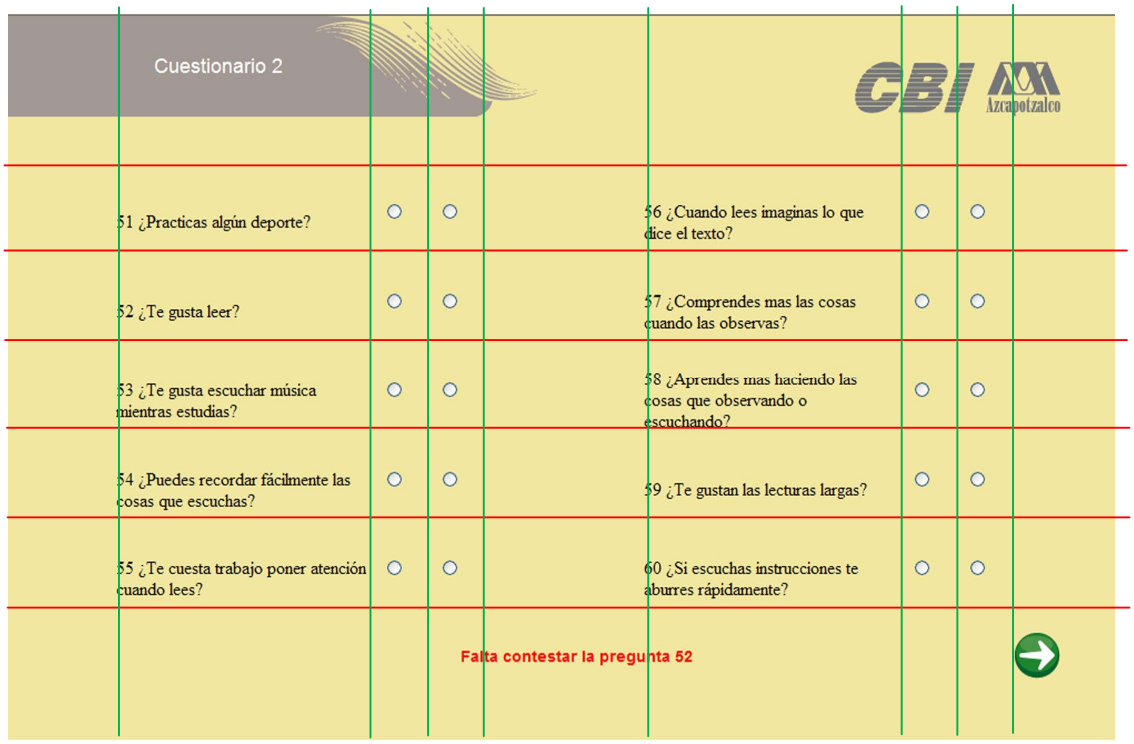


Figura A2.5 Pantalla del Cuestionario de Motivación de Orientación de Estudio del Alumno.

La estructura de la pantalla final del cuestionario es igual a la del cambio de cuestionario. (Ver Figura A2.6)

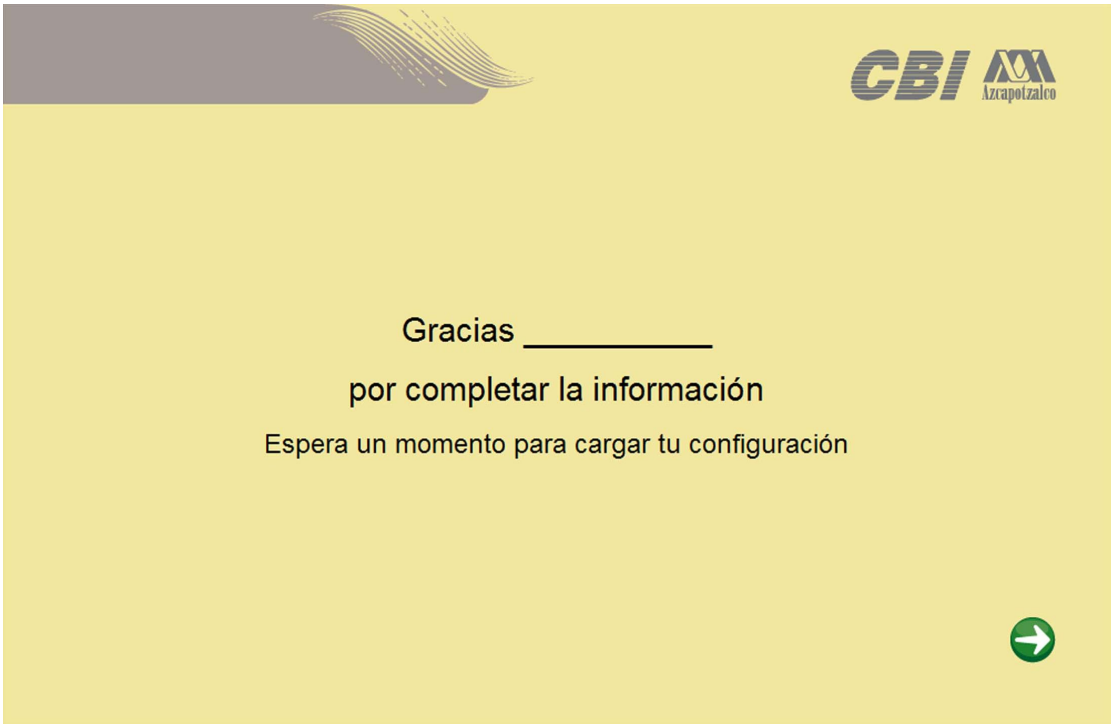


Figura A2.6 Pantalla de Resultado de la Resolución de los Cuestionarios de Estilos de Aprendizaje y de Motivación de la Orientación del Estudiante.

Retícula de Contenido

Se utiliza la misma estructura con diferentes colores dependiendo del tipo de aprendizaje de cada usuario con respecto a la psicología del color, las cuales se pueden ver en las Figuras A2.7, A28, A29 y A210.

Figura A2.7 Pantalla de la Interfaz para el Estilo de Aprendizaje Activo.

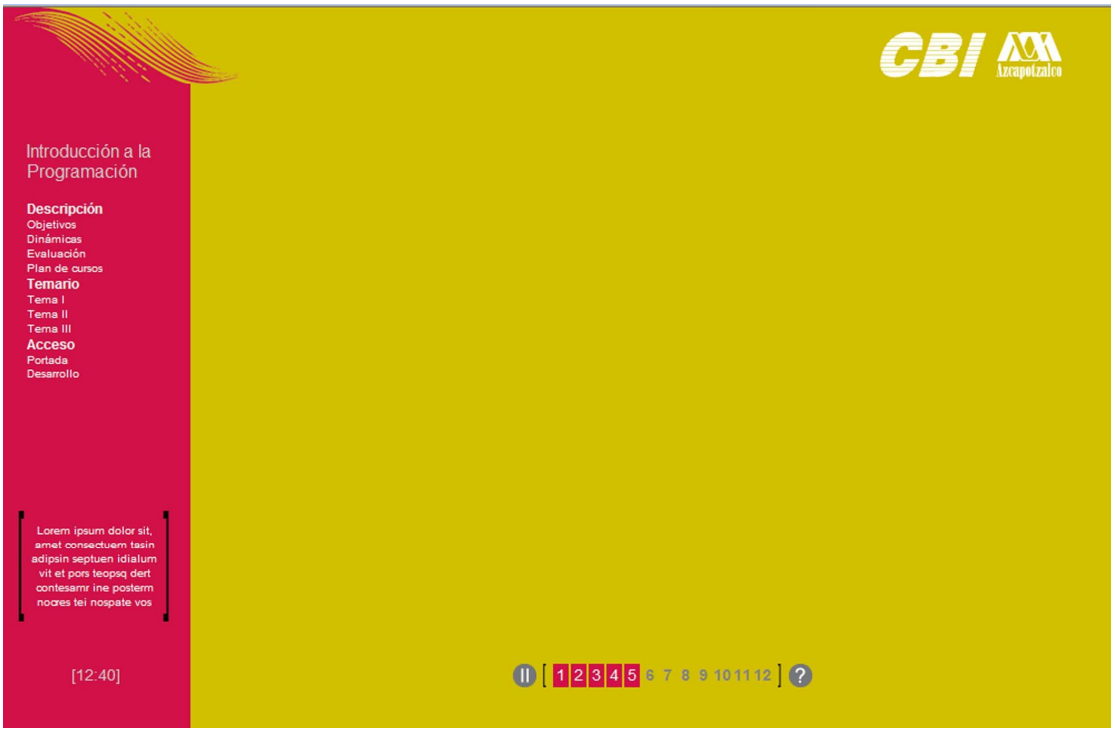


Figura A2.8 Pantalla de la Interfaz para el Estilo de Aprendizaje Pragmático.

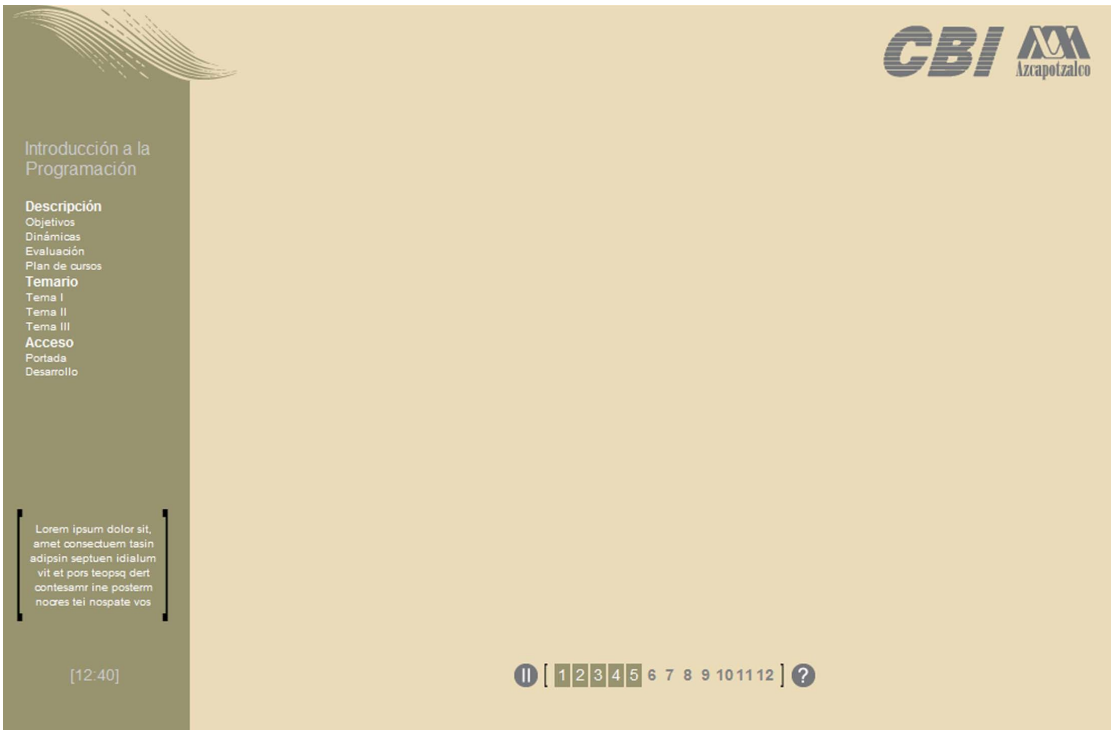


Figura A2.9 Pantalla de la Interfaz para el Estilo de Aprendizaje Reflexivo.

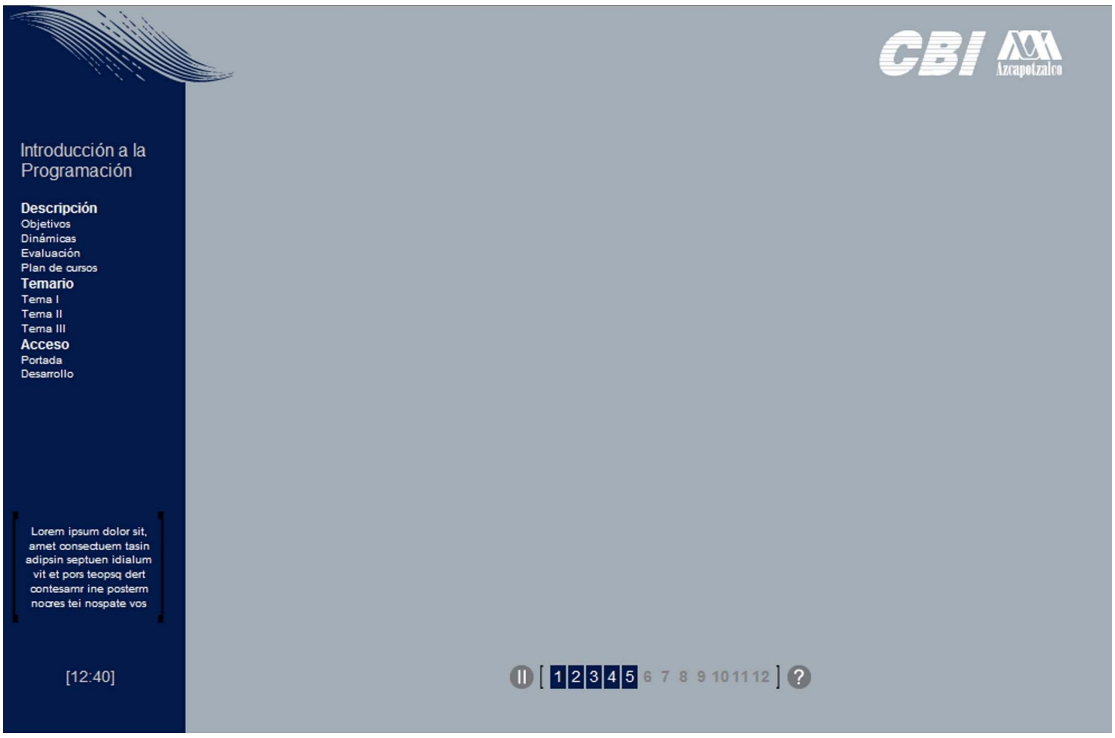
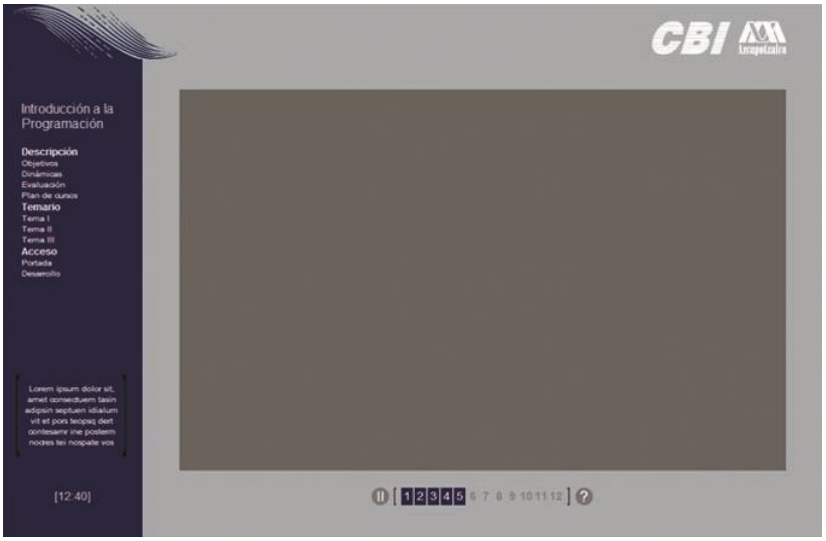
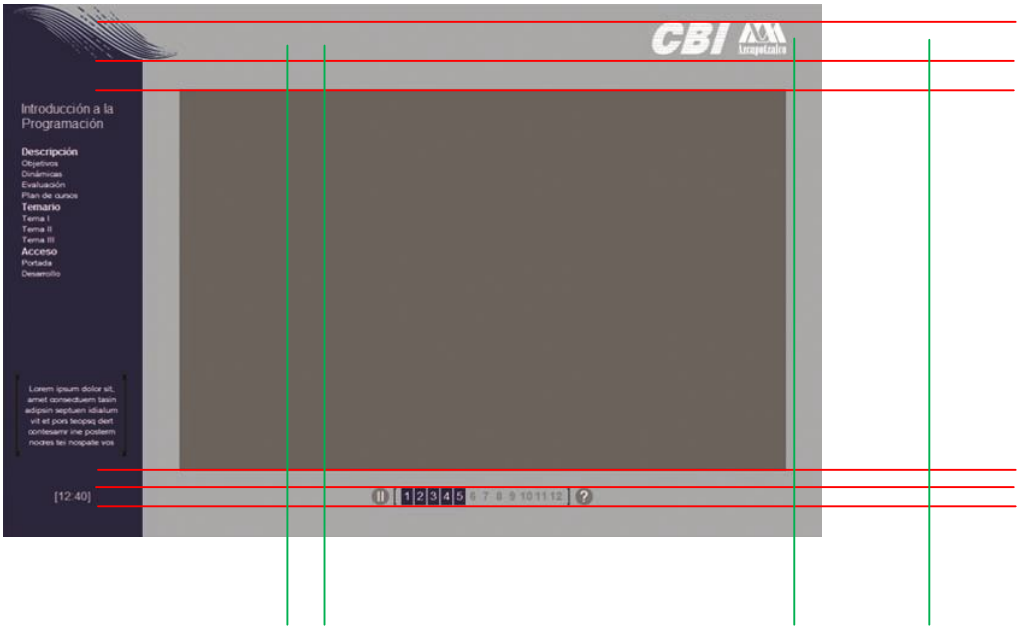


Figura A2.10 Pantalla de la Interfaz para el Estilo de Aprendizaje Teórico.

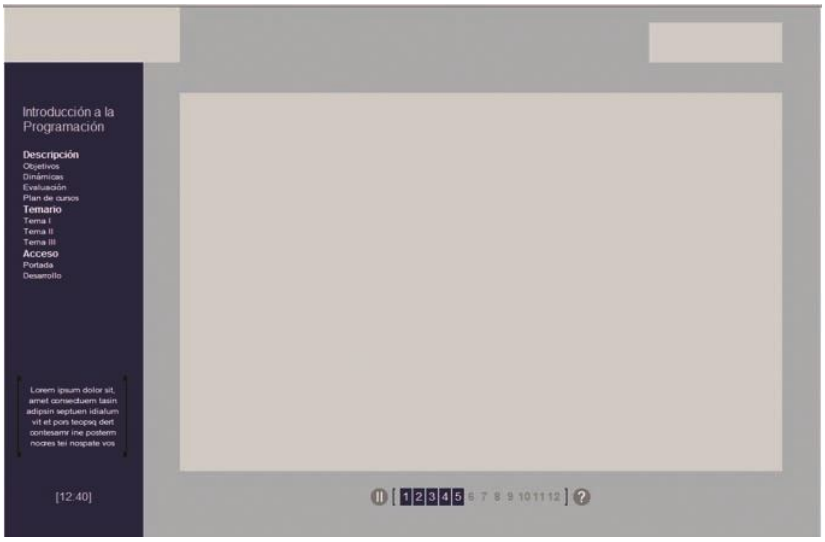
La retícula de los estilos de aprendizaje se traza en tablas con el número de celdas necesarias para el contenido.





Se inserta una tabla con el número de celdas necesarias, 6 filas y 5 columnas, en el ancho de columna se introduce el valor de la medida del sitio (1000 pixeles) en la opción Border thickness se a spacing nada al igual que en la opción Header, lo que genera una tabla con un ancho y largo de celdas proporcionales.

Todos los elementos deben estar contenidos en celdas que se colocan en los lugares establecidos por el diseño realizado para cada elemento.



Las columnas y filas de la tabla se deben ajustar a las medidas que se requieran, para ello se debe mover las líneas delimitadoras de las celdas ya sea ópticamente con las guías y reglas o métricamente seleccionando la fila o columna y cambiando su largo y/o ancho en la pestaña de propiedades con las medidas en pixeles dadas en el diseño.

Los valores de las columnas y filas se introducen en la ventana de propiedades, (situada en la parte inferior de dicha ventana) en la opción de ancho (W) para las columnas y en la opción de largo (H) para las filas.

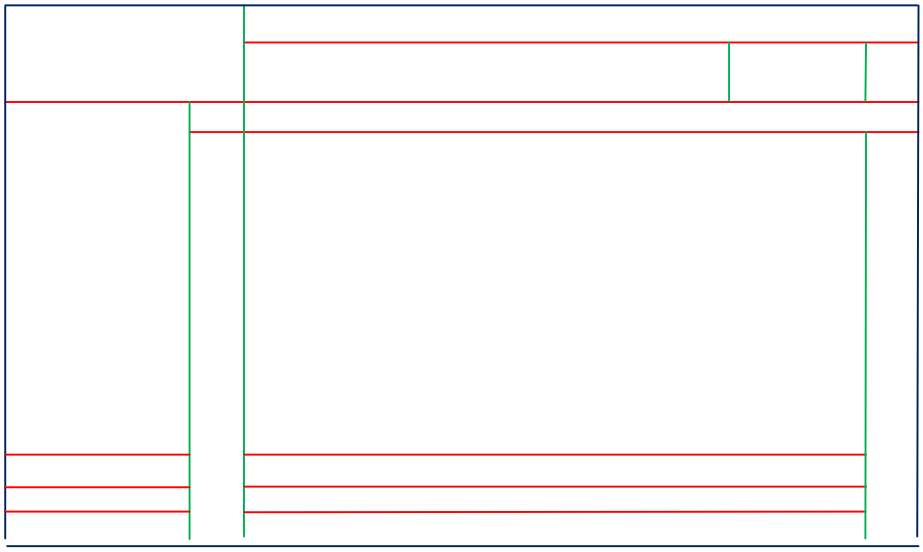
	Col 1	Col	Col 3	Col 4	Col 5
Fila 1					
Fila 2					
Fila 3					
Fila 4					
Fila 5					
Fila 6					
Fila 7					

- La columna 1 mide 170px de ancho (W=170).
- La columna 2 mide 45px de ancho (W=45).
- La columna 3 mide 575px de ancho (W=575).
- La columna 4 mide 163px de ancho (W=163).
- La columna 5 mide 47px de ancho (W=47).

- La fila 1 mide 19px de alto (H=19).
- La fila 2 mide 49px de alto (H=49).
- La fila 3 mide 37px de alto (H=37).
- La fila 4 mide 460px de alto (H=460).
- La fila 5 mide 14px de alto (H=14).
- La fila 6 mide 20px de alto (H=20).
- La fila 7 mide 38px de alto (H=38).

Por lo que al final la tabla debe medir 1000px de ancho y 650px de alto.

Después se simplifica el número de celdas uniéndolas para abarcar los espacios correspondientes al diseño.

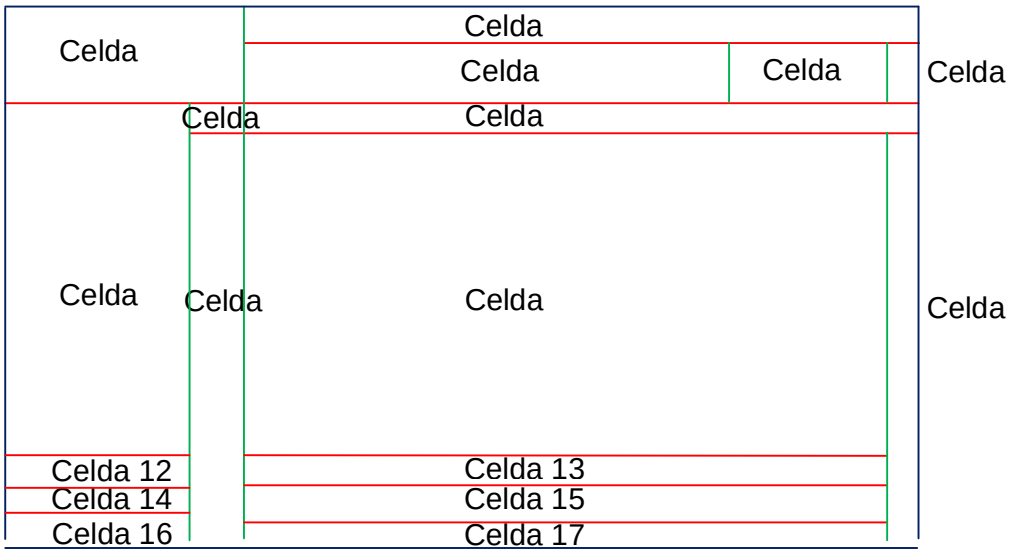


Lo que da como resultado las celdas que ya no se pueden modificar fácilmente y delimitando los espacios requeridos para imágenes, texto, cargadores, botones, links o lo que sea requerido.

Los colores de la tabla principal son dos en todos los casos, uno que es el fondo y el de menú.

- El color de fondo abarca las celdas: 2-7, 9-11, 13, 15 y 17.
- El color para el fondo activo es: #FDB813.

- El color para el fondo pragmático es: #D2BF00.
- El color para el fondo reflexivo es: #EBDCB9.
- El color para el fondo teórico es: #A2ADB5.
- Las celdas de menú abarcan: 8, 12, 14, 16.
- El color de fondo para el menú activo es: #F68B1E.
- El color de fondo para el menú pragmático es: #D11349.
- El color de fondo para el menú reflexivo es: #989470.
- El color de fondo para el menú teórico es: #001A49.



Celda 1 Copete, se coloca la imagen de copete del estilo de aprendizaje correspondiente: “activo.gif “, “pragmatico.gif”, “reflexivo.gif” y “teorico2.gif” (215px de ancho y 68px e altura)

Celda 4 Logotipos, contiene los logotipos que pueden ser de dos tipos: gris para el activo y reflexivo “logo-gris.png” (163px de ancho y 42 px de altura) blanco para el pragmático y teórico “log blanco.png” (163px de ancho y 42 px de altura).

Celda 8 Menú, se coloca en esa celda una tabla con 3 filas y 3 columnas:

- La primera columna mide 10px de ancho.

- La segunda columna mide 122px de ancho.
- La tercera columna mide 10px de ancho.
- La primera fila mide 200px de altura.
- La segunda fila mide 20px de altura.
- La tercera fila mide 100px de altura.

En la primera columna, fila 3, está la imagen “pleca-1.png” (6px de ancho y 100px de altura).

En la segunda columna, fila 1, se coloca el menú.

En la segunda columna, fila 3, se escriben las citas o el texto requerido.

En la tercera columna, fila 3, está la imagen “pleca-2.png” (6px de ancho y 100px de altura).

Celda 10 Cargador.

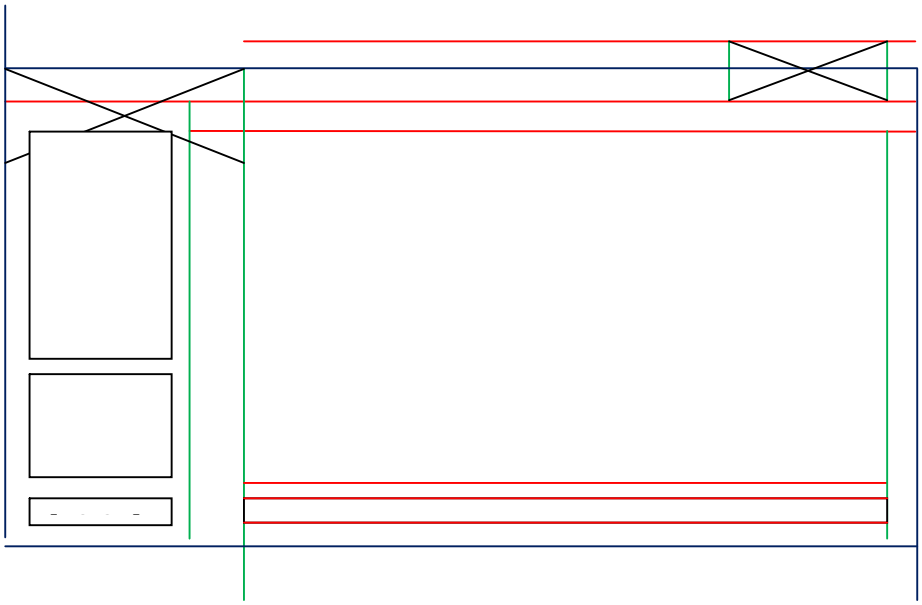
Celda 14 Reloj, es texto con justificación centrada.

Celda 15 Submenú de navegación.

Es una tabla de 16 columnas y 1 fila con 273px de ancho y 20px de altura:

Columna 1 imagen en roll over “punto.png” y “punto-sobre.png” (20px de ancho y 20 px de altura).

- Columna 2 imagen “pleca-menu-1.png” (3px de ancho y 20px de altura).
- Columna 3-14 numeración con el color de texto correspondiente y el color de fondo de la celda es el mismo que el color de fondo de celda del menú correspondiente a cada estilo de aprendizaje.
- Columna 15 imagen “pleca-menu-2.png” (3px de ancho y 20px de altura).
- Columna 16 imagen en roll over “signo.png” y “signo-sobre.png” (20px de ancho y 20px de altura).



Los colores se cambian con el selector de color para background en la ventana de propiedades.

Tablas

El fondo en las tablas se puede modificar en la pestaña de propiedades en la opción de bgc (background color) y elegir el color de la paleta o del mezclador de color, inclusive se puede colocar un fondo de imagen. (Ver Figura A2.11)

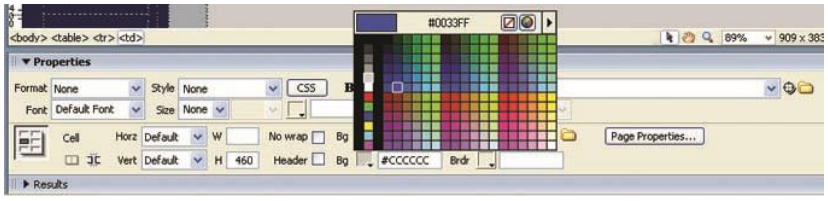


Figura A2.11 Figura de Paleta de Colores para Internet.

Si se requiere colocar un espacio entre celdas se selecciona la celda o las celdas necesarias y en la pestaña de propiedades se coloca en CellPading el número en pixeles que se requiera de espacio, ya que si se requiere poner un contorno de la misma forma y en la misma pestaña está la opción de border line para poner un margen en las celdas, si no se quiere espacio se debe poner 0.

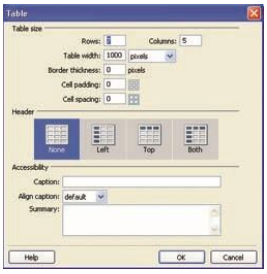


Figura A2.12 Ventana de Propiedades de una Tabla de una aplicación en Dreamweaver.

Se pueden colocar tablas dentro de celdas y también se pueden dividir las celdas en dos o más filas para evitar el sobrecargar el espacio de trabajo.

Las celdas se pueden unir para simplificar los espacios y evitar confusiones, de esta forma se pueden tener áreas de trabajo más amplias para poder introducir los contenidos.

Se debe que tener cuidado en especial con los tamaños de las celdas y los tamaños que tienen las imágenes o textos originales, ya que si éstas exceden el espacio de las celdas la tabla se desproporcionará y en la vista en web se observará el cambio que inclusive excede el espacio que ocupa una página normal.

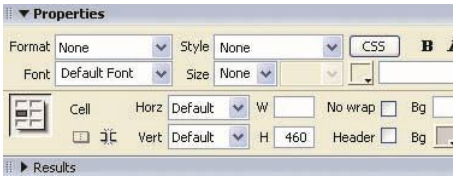


Figura A2.13 Ventana de Propiedades de una aplicación en Dreamweaver.

Dreamweaver **(Adobe Dreamweaver** es una aplicación en forma de estudio de Adobe Flash para la construcción y edición de sitios y aplicaciones Web basados en estándares, actualmente producido por Adobe Systems) permite cargar archivos de flash, pudiendo ubicar archivos y películas, así como sonido en el sitio web.

En la parte de programación se puede utilizar en modo de diseño, modo de programación o combinado, lo que permite ir observando los cambios que se aplican sobre las celdas que se están usando y familiarizarse con el lenguaje HTML y reconocer los códigos de lo que se está realizando al momento, sin necesidad de conocer o buscar los elementos en otro lado, por lo cual se

puede realizar la página escribiendo solamente código o usando el modo gráfico poniendo y realizando las acciones a la par que se visualizan y ajustan ópticamente. (Ver Figura A2.14)

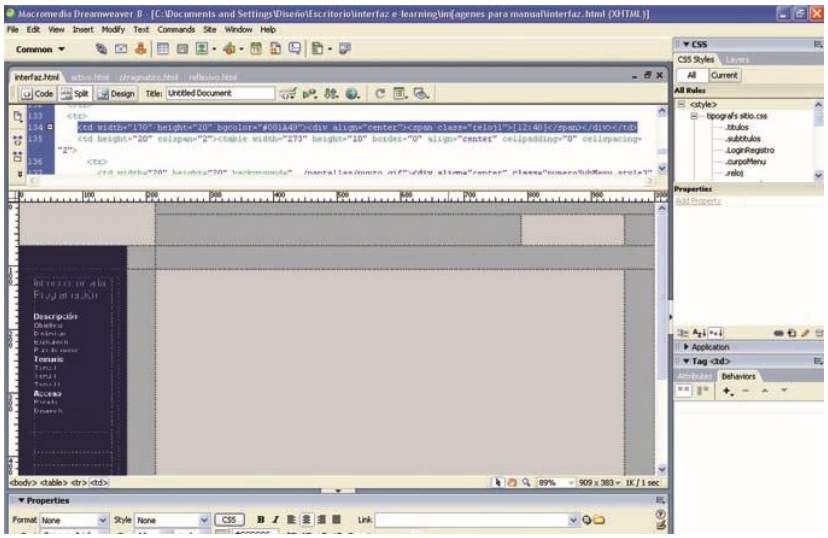


Figura A2.14 Pantalla de Inicio de una aplicación en Dreamweaver.

Estilos de Texto

La tipografía para el Login es la misma de las interfaces de contenido, una tipografía Geneva, Arial, Helvetica, sans-serif, en diferentes tamaños pata título y cuerpo texto en el cuadro de Bienvenida, y en color gris para el login.

- Color blanco: #FFFFFF
- Color gris: #333333

Se eligió la tipografía Arial, Helvetica, sans-serif para la interfaz del cuestionario, utilizando color blanco para títulos, negro para el cuerpo texto y rojo para errores.

- Color blanco: #FFFFFF
- Color negro: #000000
- Color rojo: #FF0000

Para las interfaces de contenido se eligió la tipografía: Geneva, Arial, Helvetica, sans-serif, que es una tipografía sans-serif de la familia de la Arial y Helvética de color blanco, dos tonos de gris, uno para títulos del menú y otro para el submenú de navegación.

- Color blanco: #FFFFFF
- Color gris claro: #CCCCCC
- Color gris medio: #7F7F7F
- Color gris oscuro: #838383

Los estilos son requeridos para las diferentes aplicaciones del sitio, para títulos, subtítulos, cuerpo, caracteres numéricos para el menú, etc., para este fin se elige una misma fuente en diferentes tamaños.

Para crear estilos se tiene que ir a la pestaña de propiedades y activar la casilla de CSS junto a la opción de Style, a la derecha del escenario aparece desplegada una pestaña con el nombre CSS y en la parte de abajo hay una hojita con un signo +, al hacer clic, se crea un nuevo estilo, sólo se tiene que seguir los pasos e introducir los parámetros requeridos para cada uno de los estilos que necesitan. (Ver Figura A2.15)

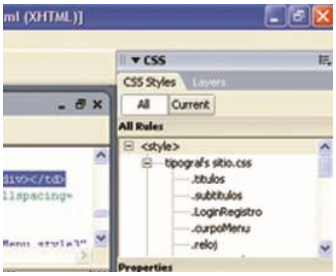


Figura A2.15 Ventana de Propiedades de una Hoja de Estilo en una aplicación de Dreamweaver.

Posteriormente se escribe el texto en la celda y se selecciona, en el menú desplegable de Style aparecerán los menús que se crearon para poder aplicarlos.

Se pueden guardar los estilos de Dreamweaver guardando el documento con

extensión .css para poder importarlo a otro documento nuevo, en el menú desplegable de la opción de Style está Attach Style Sheet y aparece el cuadro de diálogo para poder incrustar los estilos al nuevo documento. (Ver Figura A2.16)

Figura A2.16 Ventana de Propiedades para agregar una nueva Hoja de Estilo en una aplicación de Dreamweaver.

```
.titulos {
        font-family: Geneva, Arial, Helvetica, sans-serif;
        font-size: 14px;
        color: #CCCCCC;
        font-style: normal;
        line-height: normal;
        font-weight: bold;
        font-variant: normal;
        text-transform: none;
}
.subtitulos {
        font-family: Geneva, Arial, Helvetica, sans-serif;
        font-size: 12px;
        font-style: normal;
        line-height: normal;
        font-weight: bold;
        font-variant: normal;
        text-transform: none;
```

```
	color: #FFFFFF;
}
.LoginRegistro {
	font-family: Geneva, Arial, Helvetica, sans-serif;
	font-size: 14px;
	font-style: normal;
	line-height: normal;
	font-weight: normal;
	font-variant: normal;
	text-transform: none;
	color: #333333;
}
.cuerpoMenu {
	font-family: Geneva, Arial, Helvetica, sans-serif;
	font-size: 10px;
	font-style: normal;
	line-height: normal;
	font-weight: normal;
	font-variant: normal;
	text-transform: none;
	color: #FFFFFF;
}
.reloj {
	font-family: Geneva, Arial, Helvetica, sans-serif;
	font-size: 12px;
	font-style: normal;
```

```
        line-height: normal;
        font-weight: bold;
        font-variant: normal;
        text-transform: none;
        color: #CCCCCC;
}
.CuestTitulos {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 18px;
        font-style: normal;
        line-height: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
        color: #FFFFFF;
}
.CuestMensaje {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 24px;
        font-style: normal;
        line-height: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
        color: #000000;
}
```

```
.CuestMensajeBIG {
	font-family: Arial, Helvetica, sans-serif;
	font-size: 30px;
	font-style: normal;
	line-height: normal;
	font-weight: normal;
	font-variant: normal;
	text-transform: none;
}
.CuestCuerpo {
	font-family: Arial, Helvetica, sans-serif;
	font-size: 12px;
	font-style: normal;
	line-height: normal;
	font-weight: normal;
	font-variant: normal;
	text-transform: none;
	color: #000000;
}
.CuestError {
	font-family: Arial, Helvetica, sans-serif;
	font-size: 14px;
	font-style: normal;
	line-height: normal;
	font-weight: bold;
	font-variant: normal;
```

```
        text-transform: none;
        color: #FF0000;
}
.numeroSubMenu {
        font-family: Geneva, Arial, Helvetica, sans-serif;
        font-size: 14px;
        font-style: normal;
        line-height: normal;
        font-weight: bold;
        font-variant: normal;
        text-transform: lowercase;
        color: #838383;
}
.numeroSubMenuVistos {
        font-family: Geneva, Arial, Helvetica, sans-serif;
        font-size: 14px;
        font-style: normal;
        line-height: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: lowercase;
        color: #FFFFFF;
}
.LoginTitulo {
        font-family: Geneva, Arial, Helvetica, sans-serif;
        font-size: 18px;
```

```
        font-style: normal;
        line-height: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
        color: #FFFFFF;
}
.LoginCuerpo {
        font-family: Geneva, Arial, Helvetica, sans-serif;
        font-size: 12px;
        font-style: normal;
        line-height: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
        color: #FFFFFF;
}
```

Una vez creados los estilos se pueden aplicar a cualquier tipo de texto en la página, lo que mantiene una armonía en los textos y facilita su edición en Dreamweaver.

Anexo 3 "Programación del Sistema ProgEst"

En este anexo se muestra el código de la programación del Sistema ProgEst, iniciaremos por el código desarrollado para el Registro de Datos del Estudiante, siguiendo con el código de los cuestionarios de "Estilos de Aprendizaje" y el cuestionario de "Orientación de Motivación de Estudio" y el Modulo Evaluación de Escenarios del sistema.

A3.1 Programación del Registro de Datos del Estudiante

A continuación se muestra la página de Registro de datos del alumno, donde se damos de alta al usuario en la Base de Datos, como se muestra en la siguiente figura:

Código de Diseño de la Página de Registro

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Registro.aspx.cs" Inherits="Registro" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

    <title>Registro</title>

    <style type="text/css">

<!--

@import url("tipografs sitio.css");

body {

            margin-left: 0px;

            margin-top: 0px;

            margin-right: 0px;

            margin-bottom: 0px;

}

-->

</style>

<script type="text/JavaScript">

<!--

function MM_swapImgRestore() { //v3.0

  var i,x,a=document.MM_sr; for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;

}

function MM_preloadImages() { //v3.0

  var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();

    var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)

    if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
```

```
}
function MM_findObj(n, d) { //v4.01
  var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}
function MM_swapImage() { //v3.0
  var  i,j=0,x,a=MM_swapImage.arguments;  document.MM_sr=new  Array;  for(i=0;i<(a.length-
2);i+=3)
   if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!x.oSrc) x.oSrc=x.src; x.src=a[i+2];}
}
//-->
</script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <table width="1000" height="650" border="0" align="center" cellpadding="0" cellspacing="0"
bgcolor="#F2E79F">
  <tr>
    <td width="250" height="90" bgcolor="#A39994" class="CuestTitulos">
    <div align="right">Datos generales </div></td>
    <td  width="230"  height="90"><img  src="Imagenes/C-cuest1.gif"  width="230"  height="90"
alt="copete" /></td>
    <td width="285" height="90"></td>
    <td  width="185"  height="90"  align="center"  valign="bottom"><img  src="Imagenes/logo-
gris.png" width="185" height="49" alt="logos" /></td>
    <td width="50" height="90"></td>
  </tr>
```

```
<tr>
  <td height="50" colspan="5"></td>
</tr>
<tr>
  <td height="410" colspan="5">
  <table width="300" height="320" border="0" align="center" cellpadding="0" cellspacing="0">
    <tr>
      <td width="150" align="left" valign="top" class="CuestCuerpo" style="height: 30px">Nombre (s) </td>
      <td width="150" align="left" valign="top" style="height: 30px">
        <asp:TextBox ID="TxtNombre" runat="server" CausesValidation="True"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RFVNombre" runat="server" ControlToValidate="TxtNombre"
          ErrorMessage="Completa los datos"></asp:RequiredFieldValidator>
      </td>
    </tr>
     
    <tr>
      <td width="150" align="left" valign="top" class="CuestCuerpo" style="height: 29px">Apellido Paterno </td>
      <td width="150" align="left" valign="top" style="height: 29px">
        <asp:TextBox ID="TxtAPaterno" runat="server" CausesValidation="True"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RFVAPaterno" runat="server" ControlToValidate="TxtAPaterno"
          ErrorMessage="Completa los datos"></asp:RequiredFieldValidator>
      </td>
    </tr>
    <tr>
      <td width="150" height="30" align="left" valign="top" class="CuestCuerpo">Apellido Materno </td>
```

```
      <td width="150" height="30" align="left" valign="top">
        <asp:TextBox ID="TxtAMaterno" runat="server"
CausesValidation="True"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ControlToValidate="TxtAMaterno"
          ErrorMessage="Completa los datos"></asp:RequiredFieldValidator>
      </td>
    </tr>
    <tr>
      <td width="150" height="30" align="left" valign="top" class="CuestCuerpo">Sexo</td>
      <td width="150" height="30" align="left" valign="top">
        <asp:DropDownList ID="DDLSexo" runat="server" Width="160px"
CausesValidation="True">
          <asp:ListItem></asp:ListItem>
          <asp:ListItem>Femenino</asp:ListItem>
          <asp:ListItem>Masculino</asp:ListItem>
        </asp:DropDownList>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
ControlToValidate="DDLSexo"
          ErrorMessage="Completa los datos"></asp:RequiredFieldValidator>
      </td>
    </tr>
    <tr>
      <td width="150" height="30" align="left" valign="top"
class="CuestCuerpo">Nacionalidad</td>
      <td width="150" height="30" align="left" valign="top">
        <asp:TextBox ID="TxtNacionalidad" runat="server"
CausesValidation="True"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
ControlToValidate="TxtNacionalidad"
          ErrorMessage="Completa los datos"></asp:RequiredFieldValidator></td>
    </tr>
```

```
    <tr>

      <td width="150" align="left" valign="top" class="CuestCuerpo" style="height: 30px">Fecha de nacimiento </td>

      <td width="150" align="left" valign="top" style="height: 30px">

        <asp:TextBox ID="TxtFecha" runat="server" CausesValidation="True"></asp:TextBox>

        <asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server" ControlToValidate="TxtFecha"

          ErrorMessage="Completa los datos"></asp:RequiredFieldValidator></td>

    </tr>

    <tr>

      <td width="150" height="30" align="left" valign="top" class="CuestCuerpo">Matr&iacute;cula</td>

      <td width="150" height="30" align="left" valign="top">

        <asp:TextBox ID="TxtMatricula" runat="server" CausesValidation="True"></asp:TextBox>

        <asp:RequiredFieldValidator ID="RequiredFieldValidator5" runat="server" ControlToValidate="TxtMatricula"

          ErrorMessage="Completa los datos"></asp:RequiredFieldValidator></td>

    </tr>

    <tr>

      <td width="150" height="30" align="left" valign="top" class="CuestCuerpo">Carrera</td>

      <td width="150" height="30" align="left" valign="top">

        <asp:TextBox ID="TxtCarrera" runat="server" CausesValidation="True"></asp:TextBox>

        <asp:RequiredFieldValidator ID="RequiredFieldValidator6" runat="server" ControlToValidate="TxtCarrera"

          ErrorMessage="Completa los datos"></asp:RequiredFieldValidator></td>

    </tr>

  </table></td>

 </tr>

 <tr>

  <td height="50" colspan="3"><table width="500" border="0" align="right" cellpadding="0" cellspacing="0">
```

```
  <tr>
    <td align="center" style="height: 18px">
       </td>
  </tr>
</table>
</td>
    <td width="185" height="50" align="right" valign="middle">
      <asp:Button ID="CmdSiguiente" runat="server" BackColor="SeaGreen" Font-Bold="True"
          Font-Size="X-Large"  ForeColor="White"  Height="37px"  Text="→"
OnClick="CmdSiguiente_Click" />
          </td>
  </tr>
  <tr>
    <td height="50" colspan="5"></td>
  </tr>
</table>

    </div></form>
</body>
</html>
```

El código de programación de la página (Registro.aspx.cs)

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class Registro : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void CmdSiguiente_Click(object sender, EventArgs e)
    {
        int cook, aux;
        Cuestionarioip nuevo = new Cuestionarioip();

        nuevo.Nombre = TxtNombre.Text;
        nuevo.ApellidoPaterno = TxtAPaterno.Text;
        nuevo.ApellidoMaterno = TxtAMaterno.Text;
        nuevo.Sexo = DDLSexo.SelectedValue;
        nuevo.Nacionalidad = TxtNacionalidad.Text;
        nuevo.FechaNac = TxtFecha.Text;
        nuevo.Matricula = TxtMatricula.Text;
        nuevo.Carrera = TxtCarrera.Text;
        nuevo.A01 = 0;
        nuevo.A02 = 0;
        nuevo.A03 = 0;
        nuevo.A04 = 0;
        nuevo.A05 = 0;
        nuevo.A06 = 0;
        nuevo.A07 = 0;
        nuevo.A08 = 0;
```

```
nuevo.A09 = 0;
nuevo.A10 = 0;
nuevo.A11 = 0;
nuevo.A12 = 0;
nuevo.A13 = 0;
nuevo.A14 = 0;
nuevo.A15 = 0;
nuevo.A16 = 0;
nuevo.A17 = 0;
nuevo.A18 = 0;
nuevo.A19 = 0;
nuevo.A20 = 0;
nuevo.A21 = 0;
nuevo.A22 = 0;
nuevo.A23 = 0;
nuevo.A24 = 0;
nuevo.A25 = 0;
nuevo.A26 = 0;
nuevo.A27 = 0;
nuevo.A28 = 0;
nuevo.A29 = 0;
nuevo.A30 = 0;
nuevo.A31 = 0;
nuevo.A32 = 0;
nuevo.A33 = 0;
nuevo.A34 = 0;
nuevo.A35 = 0;
nuevo.A36 = 0;
nuevo.A37 = 0;
```

```
nuevo.A38 = 0;
nuevo.A39 = 0;
nuevo.A40 = 0;
nuevo.A41 = 0;
nuevo.A42 = 0;
nuevo.A43 = 0;
nuevo.A44 = 0;
nuevo.A45 = 0;
nuevo.A46 = 0;
nuevo.A47 = 0;
nuevo.A48 = 0;
nuevo.A49 = 0;
nuevo.A50 = 0;
nuevo.A51 = 0;
nuevo.A52 = 0;
nuevo.A53 = 0;
nuevo.A54 = 0;
nuevo.A55 = 0;
nuevo.A56 = 0;
nuevo.A57 = 0;
nuevo.A58 = 0;
nuevo.A59 = 0;
nuevo.A60 = 0;
nuevo.A61 = 0;
nuevo.A62 = 0;
nuevo.A63 = 0;
nuevo.A64 = 0;
nuevo.A65 = 0;
nuevo.A66 = 0;
```

```
nuevo.A67 = 0;
nuevo.A68 = 0;
nuevo.A69 = 0;
nuevo.A70 = 0;
nuevo.A71 = 0;
nuevo.A72 = 0;
nuevo.A73 = 0;
nuevo.A74 = 0;
nuevo.A75 = 0;
nuevo.A76 = 0;
nuevo.A77 = 0;
nuevo.A78 = 0;
nuevo.A79 = 0;
nuevo.A80 = 0;
nuevo.Result = "Alta";
nuevo.OrientMot01 = 0;
nuevo.OrientMot02 = 0;
nuevo.OrientMot03 = 0;
nuevo.OrientMot04 = 0;
nuevo.OrientMot05 = 0;
nuevo.OrientMot06 = 0;
nuevo.OrientMot07 = 0;
nuevo.OrientMot08 = 0;
nuevo.Resultado = "EnProceso";
aux=nuevo.AltaRegistro();
if (aux == 1)
{
    Cuestionarioip galleta = new Cuestionarioip();
    galleta.Result="Alta";
```

```
            cook = galleta.LecturaID();
            Response.Cookies["UserID"].Value = Convert.ToString(cook);
            Response.Redirect("Chaea01.aspx");
        }
        else
        {
            Server.Transfer("Error.aspx");
        }
    }
}
```

A3.2 Programación del Cuestionario "HONEY-ALONSO de Estilos de Aprendizaje"

El cuestionario esta realizado en ASP utilizando C# como lenguaje de programación.

El sitio Web conformado de 13 páginas en ASPX, y como Base de Datos del sistema ProgEst se instrumento en SQL Server. Para la hoja de estilo un archivo denominado de hoja de estilo, una página Chaea.cs (como objeto que se utiliza para la programación del sitio a la base de datos).

En el siguiente diagrama se muestra cómo funciona el sitio Web:

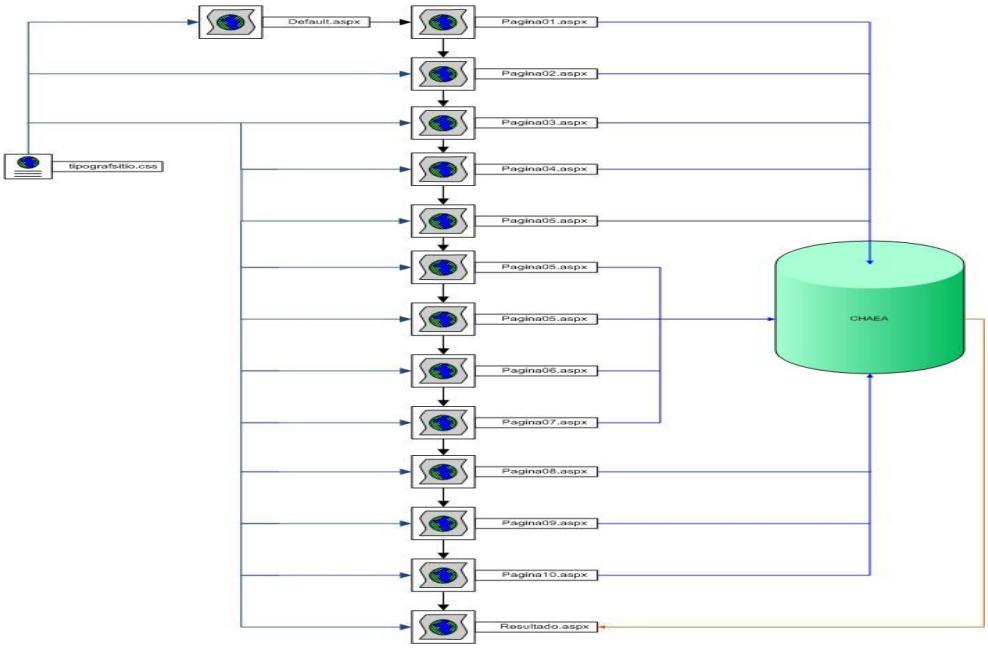


Figura A3.1 Sitio Web CHAEA.

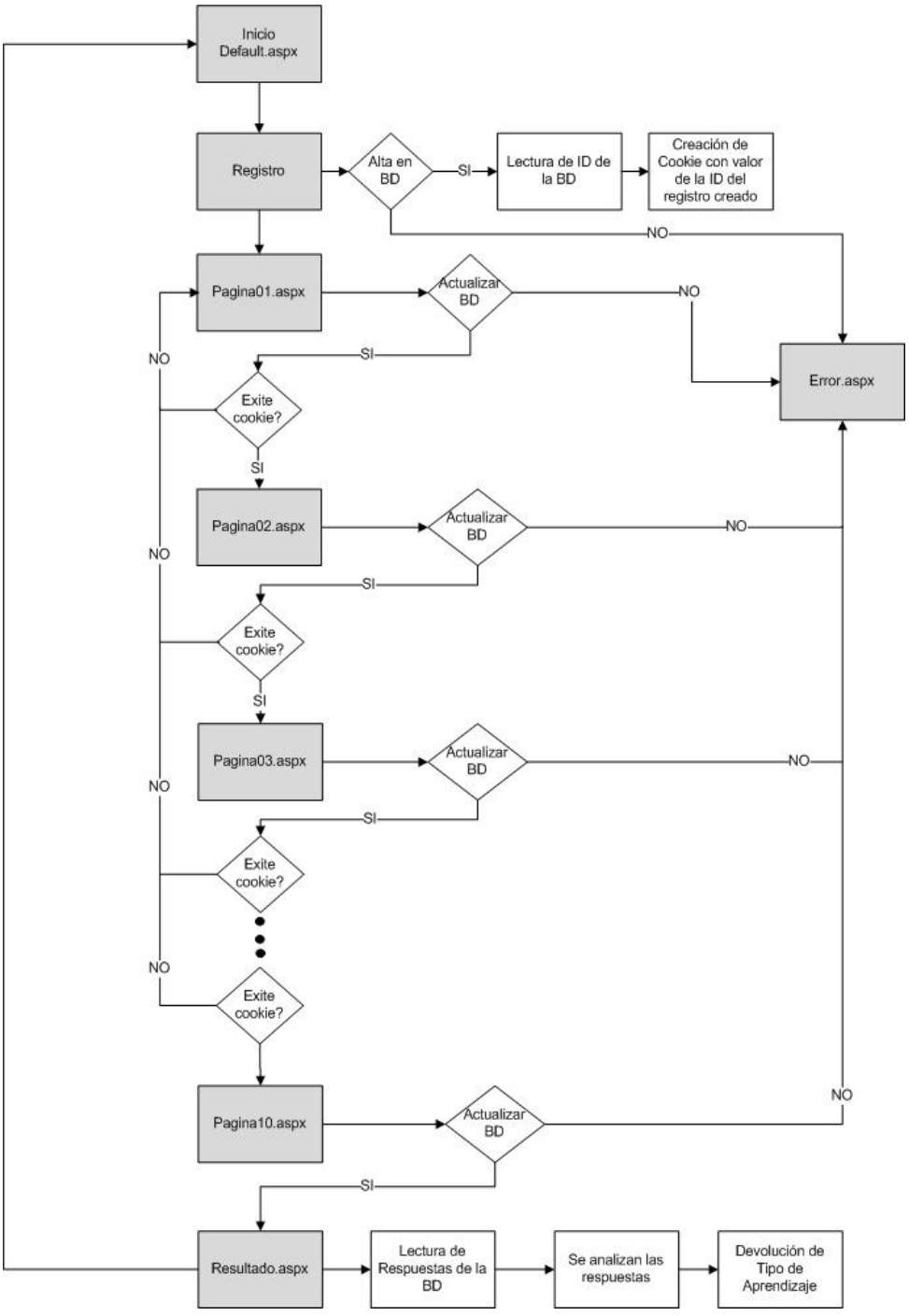


Figura A3.2 Diagrama de Flujo de la Aplicación Web.

En la Figura 2 se observa el diagrama de flujo de la aplicación en donde la pagina principal (default.aspx), contiene la opción de Registro y de Login, para los casos donde los usuarios de sistema que ya están registrados en el sistema.

La primera página del cuestionario (Registro.aspx) es la más importante de todas, dado que en esta página se crea el registro en la base de datos que vamos a usar en el cuestionario, además se crea una cookie con la ID del registro con el fin de que en las páginas siguientes sea nuestra referencia para actualizar nuestra base de datos.

En las páginas siguientes se verifica que la cookie exista y lea el valor, con este valor de ID del Registro realizaremos una actualización a nuestra base de datos, en caso de que no exista la cookie regresaremos a la primera pagina del cuestionario para que se cree. Esto se realiza hasta que se contesten todas las preguntas del cuestionario.

En la última página del cuestionario (Resultado.aspx) es donde se realiza el análisis de las respuestas capturadas en la base de datos y así obtener el resultado de nuestra encuesta.

En la Figura 1 podemos observar que la página principal (maestra) afecta a todas las páginas ASPX, una página maestra nos permite crear un diseño coherente para las páginas, definimos el aspecto, el diseño y comportamiento básico que van a tener todas las páginas de aplicación en esta página, esta página maestra utiliza el archivo *tipografsitio.css*, que es un archivo de hoja de estilos.

Códigos de la Aplicación Web

Código archivo tipografsitio.css

```
.titulos {
        font-family: Geneva, Arial, Helvetica, sans-serif;
        font-size: 16px;
        color: #7F7F7F;
        font-style: normal;
        line-height: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
```

```
}
.subtitulos {
	font-family: Geneva, Arial, Helvetica, sans-serif;
	font-size: 12px;
	font-style: normal;
	line-height: normal;
	font-weight: bold;
	font-variant: normal;
	text-transform: none;
	color: #FFFFFF;
}
.curpoMenu {
	font-family: Geneva, Arial, Helvetica, sans-serif;
	font-size: 10px;
	font-style: normal;
	line-height: normal;
	font-weight: normal;
	font-variant: normal;
	text-transform: none;
	color: #FFFFFF;
}
.reloj {
	font-family: Geneva, Arial, Helvetica, sans-serif;
	font-size: 14px;
	font-style: normal;
	line-height: normal;
	font-weight: normal;
	font-variant: normal;
	text-transform: none;
```

```
        color: #838383;
}
.numeroSubMenu {
        font-family: Geneva, Arial, Helvetica, sans-serif;
        font-size: 14px;
        font-style: normal;
        line-height: normal;
        font-weight: bold;
        font-variant: normal;
        text-transform: lowercase;
        color: #838383;
}
.numeroSubMenuVistos {
        font-family: Geneva, Arial, Helvetica, sans-serif;
        font-size: 14px;
        font-style: normal;
        line-height: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: lowercase;
        color: #FFFFFF;
}
```

A continuación se muestra la pantalla de presentación del cuestionario de Estilos de Aprendizaje.

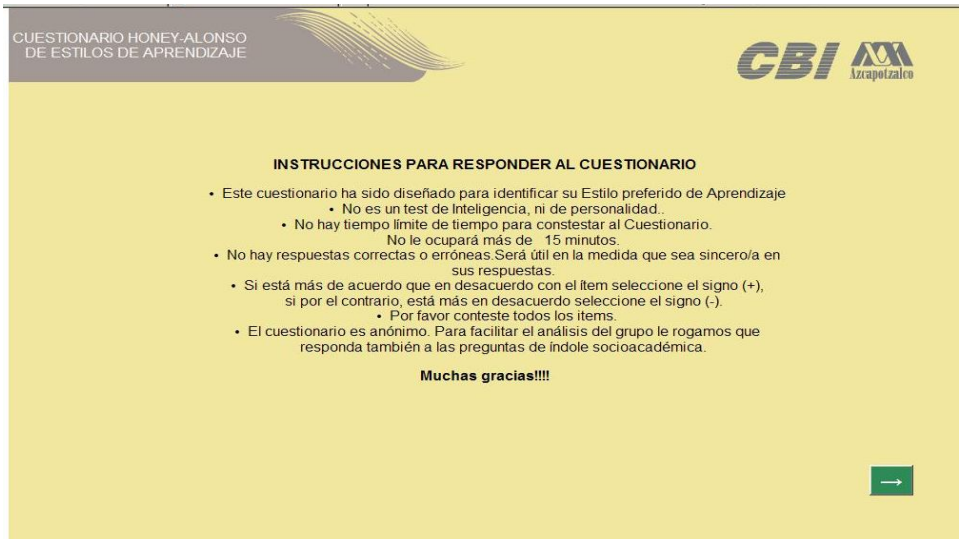


Figura A3.3 Aspecto de la página principal del Cuestionario de Estilos de Aprendizaje

Objeto Cuestionario

El Objeto Cuestionario está constituido por dos partes, la primera parte es para el cuestionario Chaea y la segunda parte es para el cuestionario de motivación de estudio. Contiene todos los métodos que se utilizan para escribir a la base de datos, así como las propiedades que van a ser utilizadas por el objeto.

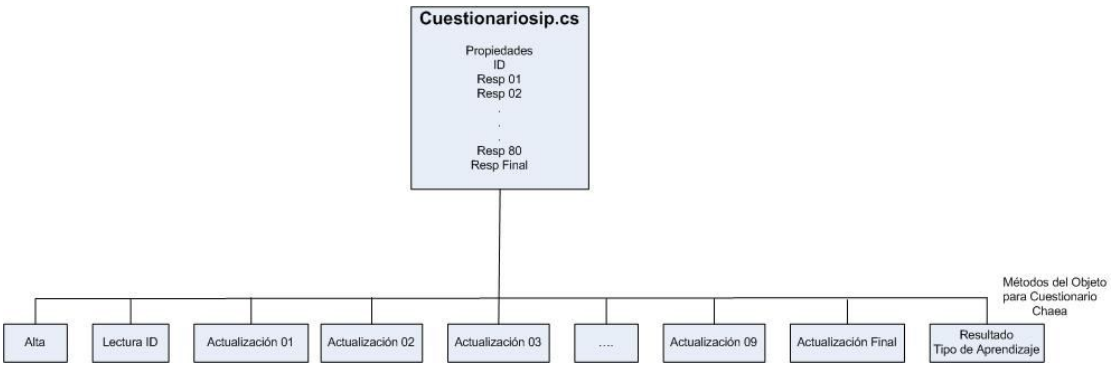


Figura A3.4 Objetos CHAEA con sus métodos utilizados en la aplicación.

Código del Cuestionario de Estilos de Aprendizaje (CHAEA)

```
// Librerias utilizadas
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

/// <summary>
/// Descripción breve de Cuestionarioip
/// Objeto para realizar las operaciones en la base de datos
/// CHAEA en su registro Cuestionarios
/// </summary>
public class Cuestionarioip
{ //Inicio de la Clase Cuestionarioip
    //Declaracion de variables para la establecer conexion a la
    //Base de datos
    private string mStrConexion;
    private SqlConnection mObjConexion;

    //Constructor
    public Cuestionarioip()
        {
```

```
    mStrConexion = "server=PROSOFT02;uid=Saigti;pwd=xhunaxi;database=CHAEA";
    mObjConexion = new SqlConnection(mStrConexion);
    mObjConexion.Open();
        }

 //Propiedades de nuestro Objeto Cuestionarioip
 private int _ID;
 public int ID
 {
    get { return _ID; }
    set { _ID = value; }
 }
//Parte de las propiedades que se utilizará para el Registro

 private string _Nombre;
 public string Nombre
 {
    get { return _Nombre; }
    set { _Nombre = value; }
 }

 private string _ApellidoPaterno;
 public string ApellidoPaterno
 {
    get { return _ApellidoPaterno; }
    set { _ApellidoPaterno = value; }
 }

 private string _ApellidoMaterno;
```

```
public string ApellidoMaterno
{
    get { return _ApellidoMaterno; }
    set { _ApellidoMaterno = value; }
}

private string _Sexo;
public string Sexo
{
    get { return _Sexo; }
    set { _Sexo = value; }
}

private string _Nacionalidad;
public string Nacionalidad
{
    get { return _Nacionalidad; }
    set { _Nacionalidad = value; }
}

private string _FechaNac;
public string FechaNac
{
    get { return _FechaNac; }
    set { _FechaNac = value; }
}

private string _Matricula;
public string Matricula
```

```
{
    get { return _Matricula; }
    set { _Matricula = value; }
}

private string _Carrera;
public string Carrera
{
    get { return _Carrera; }
    set { _Carrera = value; }
}

//Parte de las propiedades utilizada para llenar el cuestionario CHAEA
private int _A01;
public int A01
{
    get { return _A01; }
    set { _A01 = value; }
}

private int _A02;
public int A02
{
    get { return _A02; }
    set { _A02 = value; }
}

private int _A03;
public int A03
```

```
{
    get { return _A03; }
    set { _A03 = value; }
}

private int _A04;
public int A04
{
    get { return _A04; }
    set { _A04 = value; }
}

private int _A05;
public int A05
{
    get { return _A05; }
    set { _A05 = value; }
}

private int _A06;
public int A06
{
    get { return _A06; }
    set { _A06 = value; }
}

private int _A07;
public int A07
{
```

```
    get { return _A07; }
    set { _A07 = value; }
}

private int _A08;
public int A08
{
    get { return _A08; }
    set { _A08 = value; }
}

private int _A09;
public int A09
{
    get { return _A09; }
    set { _A09 = value; }
}

private int _A10;
public int A10
{
    get { return _A10; }
    set { _A10 = value; }
}


private int _A11;
public int A11
{
```

```
    get { return _A11; }
    set { _A11 = value; }
}

private int _A12;
public int A12
{
    get { return _A12; }
    set { _A12 = value; }
}

private int _A13;
public int A13
{
    get { return _A13; }
    set { _A13 = value; }
}

private int _A14;
public int A14
{
    get { return _A14; }
    set { _A14 = value; }
}

private int _A15;
public int A15
{
    get { return _A15; }
```

```
    set { _A15 = value; }
}

private int _A16;
public int A16
{
    get { return _A16; }
    set { _A16 = value; }
}

private int _A17;
public int A17
{
    get { return _A17; }
    set { _A17 = value; }
}

private int _A18;
public int A18
{
    get { return _A18; }
    set { _A18 = value; }
}

private int _A19;
public int A19
{
    get { return _A19; }
    set { _A19 = value; }
```

```
}

private int _A20;
public int A20
{
    get { return _A20; }
    set { _A20 = value; }
}

private int _A21;
public int A21
{
    get { return _A21; }
    set { _A21 = value; }
}

private int _A22;

public int A22
{
    get { return _A22; }
    set { _A22 = value; }
}

private int _A23;
public int A23
{
    get { return _A23; }
    set { _A23 = value; }
```

```
}

private int _A24;
public int A24
{
    get { return _A24; }
    set { _A24 = value; }
}

private int _A25;
public int A25
{
    get { return _A25; }
    set { _A25 = value; }
}

private int _A26;
public int A26
{
    get { return _A26; }
    set { _A26 = value; }
}

private int _A27;
public int A27
{
    get { return _A27; }
    set { _A27 = value; }
}
```

```
private int _A28;
public int A28
{
    get { return _A28; }
    set { _A28 = value; }
}

private int _A29;
public int A29
{
    get { return _A29; }
    set { _A29 = value; }
}

private int _A30;
public int A30
{
    get { return _A30; }
    set { _A30 = value; }
}

private int _A31;
public int A31
{
    get { return _A31; }
    set { _A31 = value; }
}
```

```
private int _A32;
public int A32
{
    get { return _A32; }
    set { _A32 = value; }
}

private int _A33;

public int A33
{
    get { return _A33; }
    set { _A33 = value; }
}

private int _A34;
public int A34
{
    get { return _A34; }
    set { _A34 = value; }
}

private int _A35;
public int A35
{
    get { return _A35; }
    set { _A35 = value; }
}
```

```
private int _A36;
public int A36
{
    get { return _A36; }
    set { _A36 = value; }
}

private int _A37;
public int A37
{
    get { return _A37; }
    set { _A37 = value; }
}

private int _A38;
public int A38
{
    get { return _A38; }
    set { _A38 = value; }
}

private int _A39;
public int A39
{
    get { return _A39; }
    set { _A39 = value; }
}

private int _A40;
```

```
public int A40
{
    get { return _A40; }
    set { _A40 = value; }
}

private int _A41;
public int A41
{
    get { return _A41; }
    set { _A41 = value; }
}

private int _A42;
public int A42
{
    get { return _A42; }
    set { _A42 = value; }
}

private int _A43;
public int A43
{
    get { return _A43; }
    set { _A43 = value; }
}

private int _A44;
```

```
public int A44
{
    get { return _A44; }
    set { _A44 = value; }
}

private int _A45;
public int A45
{
    get { return _A45; }
    set { _A45 = value; }
}

private int _A46;
public int A46
{
    get { return _A46; }
    set { _A46 = value; }
}

private int _A47;
public int A47
{
    get { return _A47; }
    set { _A47 = value; }
}

private int _A48;
public int A48
```

```
{
    get { return _A48; }
    set { _A48 = value; }
}

private int _A49;
public int A49
{
    get { return _A49; }
    set { _A49 = value; }
}

private int _A50;
public int A50
{
    get { return _A50; }
    set { _A50 = value; }
}


private int _A51;
public int A51
{
    get { return _A51; }
    set { _A51 = value; }
}

private int _A52;
public int A52
```

```
{
    get { return _A52; }
    set { _A52 = value; }
}

private int _A53;
public int A53
{
    get { return _A53; }
    set { _A53 = value; }
}

private int _A54;
public int A54
{
    get { return _A54; }
    set { _A54 = value; }
}

private int _A55;
public int A55
{
    get { return _A55; }
    set { _A55 = value; }
}

private int _A56;
public int A56
{
```

```
    get { return _A56; }
    set { _A56 = value; }
}

private int _A57;
public int A57
{
    get { return _A57; }
    set { _A57 = value; }
}

private int _A58;
public int A58
{
    get { return _A58; }
    set { _A58 = value; }
}

private int _A59;
public int A59
{
    get { return _A59; }
    set { _A59 = value; }
}

private int _A60;
public int A60
{
    get { return _A60; }
```

```
    set { _A60 = value; }
}

private int _A61;
public int A61
{
    get { return _A61; }
    set { _A61 = value; }
}

private int _A62;

public int A62
{
    get { return _A62; }
    set { _A62 = value; }
}

private int _A63;
public int A63
{
    get { return _A63; }
    set { _A63 = value; }
}

private int _A64;
public int A64
{
    get { return _A64; }
```

```
    set { _A64 = value; }
}

private int _A65;
public int A65
{
    get { return _A65; }
    set { _A65 = value; }
}

private int _A66;
public int A66
{
    get { return _A66; }
    set { _A66 = value; }
}

private int _A67;
public int A67
{
    get { return _A67; }
    set { _A67 = value; }
}

private int _A68;
public int A68
{
    get { return _A68; }
    set { _A68 = value; }
```

```
}

private int _A69;
public int A69
{
    get { return _A69; }
    set { _A69 = value; }
}

private int _A70;
public int A70
{
    get { return _A70; }
    set { _A70 = value; }
}


private int _A71;
public int A71
{
    get { return _A71; }
    set { _A71 = value; }
}

private int _A72;
public int A72
{
    get { return _A72; }
    set { _A72 = value; }
```

```
}

private int _A73;
public int A73
{
    get { return _A73; }
    set { _A73 = value; }
}

private int _A74;
public int A74
{
    get { return _A74; }
    set { _A74 = value; }
}

private int _A75;
public int A75
{
    get { return _A75; }
    set { _A75 = value; }
}

private int _A76;
public int A76
{
    get { return _A76; }
    set { _A76 = value; }
}
```

```
private int _A77;
public int A77
{
    get { return _A77; }
    set { _A77 = value; }
}

private int _A78;
public int A78
{
    get { return _A78; }
    set { _A78 = value; }
}

private int _A79;
public int A79
{
    get { return _A79; }
    set { _A79 = value; }
}

private int _A80;
public int A80
{
    get { return _A80; }
    set { _A80 = value; }
}
```

```
private string _Result;
public string Result
{
    get { return _Result; }
    set { _Result = value; }
}

//Parte de las propiedades para el cuestionario de Orientacion
private int _OrientMot01;
public int OrientMot01
{
    get { return _OrientMot01; }
    set { _OrientMot01 = value; }
}
private int _OrientMot02;
public int OrientMot02
{
    get { return _OrientMot02; }
    set { _OrientMot02 = value; }
}

private int _OrientMot03;
public int OrientMot03
{
    get { return _OrientMot03; }
    set { _OrientMot03 = value; }
}
private int _OrientMot04;
public int OrientMot04
```

```
{
    get { return _OrientMot04; }
    set { _OrientMot04 = value; }
}
private int _OrientMot05;
public int OrientMot05
{
    get { return _OrientMot05; }
    set { _OrientMot05 = value; }
}
private int _OrientMot06;
public int OrientMot06
{
    get { return _OrientMot06; }
    set { _OrientMot06 = value; }
}
private int _OrientMot07;
public int OrientMot07
{
    get { return _OrientMot07; }
    set { _OrientMot07 = value; }
}
private int _OrientMot08;
public int OrientMot08
{
    get { return _OrientMot08; }
    set { _OrientMot08 = value; }
}
```

```
    private string _Resultado;
    public string Resultado
    {
        get { return _Resultado; }
        set { _Resultado = value; }
    }

    //Fin de las propiedades del objeto
    //Metodos del Objeto
    //Alta del Registro
    public int AltaRegistro()
    {
        //SqlDataReader objDataReader;
        string CmdAlta = "INSERT INTO Cuestionarios
VALUES(@Nombre,@ApellidoPaterno,@ApellidoMaterno,@Sexo,@Nacionalidad,@FechaNac,
@Matricula,@Carrera,@Resp01,@Resp02,@Resp03,@Resp04,@Resp05,@Resp06,@Resp0
7,@Resp08,@Resp09,@Resp10,@Resp11,@Resp12,@Resp13,@Resp14,@Resp15,@Resp
16,@Resp17,@Resp18,@Resp19,@Resp20,@Resp21,@Resp22,@Resp23,@Resp24,@Res
p25,@Resp26,@Resp27,@Resp28,@Resp29,@Resp30,@Resp31,@Resp32,@Resp33,@Re
sp34,@Resp35,@Resp36,@Resp37,@Resp38,@Resp39,@Resp40,@Resp41,@Resp42,@R
esp43,@Resp44,@Resp45,@Resp46,@Resp47,@Resp48,@Resp49,@Resp50,@Resp51,@
Resp52,@Resp53,@Resp54,@Resp55,@Resp56,@Resp57,@Resp58,@Resp59,@Resp60,
@Resp61,@Resp62,@Resp63,@Resp64,@Resp65,@Resp66,@Resp67,@Resp68,@Resp69
,@Resp70,@Resp71,@Resp72,@Resp73,@Resp74,@Resp75,@Resp76,@Resp77,@Resp7
8,@Resp79,@Resp80,@RespFin,@OrientMotR1,@OrientMotR2,@OrientMotR3,@OrientMotR
4,@OrientMotR5,@OrientMotR6,@OrientMotR7,@OrientMotR8,@Resultado)";
        SqlCommand sqlAlta = new SqlCommand(CmdAlta, mObjConexion);
        //Alta Registro
        sqlAlta.Parameters.Add(new SqlParameter("@Nombre", SqlDbType.NVarChar));
        sqlAlta.Parameters["@Nombre"].Value = _Nombre;
        sqlAlta.Parameters.Add(new SqlParameter("@ApellidoPaterno", SqlDbType.NVarChar));
        sqlAlta.Parameters["@ApellidoPaterno"].Value = _ApellidoPaterno;
        sqlAlta.Parameters.Add(new SqlParameter("@ApellidoMaterno", SqlDbType.NVarChar));
        sqlAlta.Parameters["@ApellidoMaterno"].Value = _ApellidoMaterno;
        sqlAlta.Parameters.Add(new SqlParameter("@Sexo", SqlDbType.NVarChar));
```

```
sqlAlta.Parameters["@Sexo"].Value = _Sexo;
sqlAlta.Parameters.Add(new SqlParameter("@Nacionalidad", SqlDbType.NVarChar));
sqlAlta.Parameters["@Nacionalidad"].Value = _Nacionalidad;
sqlAlta.Parameters.Add(new SqlParameter("@FechaNac", SqlDbType.NVarChar));
sqlAlta.Parameters["@FechaNac"].Value = _FechaNac;
sqlAlta.Parameters.Add(new SqlParameter("@Matricula", SqlDbType.NVarChar));
sqlAlta.Parameters["@Matricula"].Value = _Matricula;
sqlAlta.Parameters.Add(new SqlParameter("@Carrera", SqlDbType.NVarChar));
sqlAlta.Parameters["@Carrera"].Value = _Carrera;

//Alta cuestionario CHAEA
sqlAlta.Parameters.Add(new SqlParameter("@Resp01", SqlDbType.Int));
sqlAlta.Parameters["@Resp01"].Value = _A01;
sqlAlta.Parameters.Add(new SqlParameter("@Resp02", SqlDbType.Int));
sqlAlta.Parameters["@Resp02"].Value = _A02;
sqlAlta.Parameters.Add(new SqlParameter("@Resp03", SqlDbType.Int));
sqlAlta.Parameters["@Resp03"].Value = _A03;
sqlAlta.Parameters.Add(new SqlParameter("@Resp04", SqlDbType.Int));
sqlAlta.Parameters["@Resp04"].Value = _A04;
sqlAlta.Parameters.Add(new SqlParameter("@Resp05", SqlDbType.Int));
sqlAlta.Parameters["@Resp05"].Value = _A05;
sqlAlta.Parameters.Add(new SqlParameter("@Resp06", SqlDbType.Int));
sqlAlta.Parameters["@Resp06"].Value = _A06;
sqlAlta.Parameters.Add(new SqlParameter("@Resp07", SqlDbType.Int));
sqlAlta.Parameters["@Resp07"].Value = _A07;
sqlAlta.Parameters.Add(new SqlParameter("@Resp08", SqlDbType.Int));
sqlAlta.Parameters["@Resp08"].Value = _A08;
sqlAlta.Parameters.Add(new SqlParameter("@Resp09", SqlDbType.Int));
sqlAlta.Parameters["@Resp09"].Value = _A09;
```

```
sqlAlta.Parameters.Add(new SqlParameter("@Resp10", SqlDbType.Int));
sqlAlta.Parameters["@Resp10"].Value = _A10;

sqlAlta.Parameters.Add(new SqlParameter("@Resp11", SqlDbType.Int));
sqlAlta.Parameters["@Resp11"].Value = _A11;
sqlAlta.Parameters.Add(new SqlParameter("@Resp12", SqlDbType.Int));
sqlAlta.Parameters["@Resp12"].Value = _A12;
sqlAlta.Parameters.Add(new SqlParameter("@Resp13", SqlDbType.Int));
sqlAlta.Parameters["@Resp13"].Value = _A13;
sqlAlta.Parameters.Add(new SqlParameter("@Resp14", SqlDbType.Int));
sqlAlta.Parameters["@Resp14"].Value = _A14;
sqlAlta.Parameters.Add(new SqlParameter("@Resp15", SqlDbType.Int));
sqlAlta.Parameters["@Resp15"].Value = _A15;
sqlAlta.Parameters.Add(new SqlParameter("@Resp16", SqlDbType.Int));
sqlAlta.Parameters["@Resp16"].Value = _A16;
sqlAlta.Parameters.Add(new SqlParameter("@Resp17", SqlDbType.Int));
sqlAlta.Parameters["@Resp17"].Value = _A17;
sqlAlta.Parameters.Add(new SqlParameter("@Resp18", SqlDbType.Int));
sqlAlta.Parameters["@Resp18"].Value = _A18;
sqlAlta.Parameters.Add(new SqlParameter("@Resp19", SqlDbType.Int));
sqlAlta.Parameters["@Resp19"].Value = _A19;
sqlAlta.Parameters.Add(new SqlParameter("@Resp20", SqlDbType.Int));
sqlAlta.Parameters["@Resp20"].Value = _A20;

sqlAlta.Parameters.Add(new SqlParameter("@Resp21", SqlDbType.Int));
sqlAlta.Parameters["@Resp21"].Value = _A21;
sqlAlta.Parameters.Add(new SqlParameter("@Resp22", SqlDbType.Int));
sqlAlta.Parameters["@Resp22"].Value = _A22;
sqlAlta.Parameters.Add(new SqlParameter("@Resp23", SqlDbType.Int));
```

```
sqlAlta.Parameters["@Resp23"].Value = _A23;
sqlAlta.Parameters.Add(new SqlParameter("@Resp24", SqlDbType.Int));
sqlAlta.Parameters["@Resp24"].Value = _A24;
sqlAlta.Parameters.Add(new SqlParameter("@Resp25", SqlDbType.Int));
sqlAlta.Parameters["@Resp25"].Value = _A25;
sqlAlta.Parameters.Add(new SqlParameter("@Resp26", SqlDbType.Int));
sqlAlta.Parameters["@Resp26"].Value = _A26;
sqlAlta.Parameters.Add(new SqlParameter("@Resp27", SqlDbType.Int));
sqlAlta.Parameters["@Resp27"].Value = _A27;
sqlAlta.Parameters.Add(new SqlParameter("@Resp28", SqlDbType.Int));
sqlAlta.Parameters["@Resp28"].Value = _A28;
sqlAlta.Parameters.Add(new SqlParameter("@Resp29", SqlDbType.Int));
sqlAlta.Parameters["@Resp29"].Value = _A29;
sqlAlta.Parameters.Add(new SqlParameter("@Resp30", SqlDbType.Int));
sqlAlta.Parameters["@Resp30"].Value = _A30;

sqlAlta.Parameters.Add(new SqlParameter("@Resp31", SqlDbType.Int));
sqlAlta.Parameters["@Resp31"].Value = _A31;
sqlAlta.Parameters.Add(new SqlParameter("@Resp32", SqlDbType.Int));
sqlAlta.Parameters["@Resp32"].Value = _A32;
sqlAlta.Parameters.Add(new SqlParameter("@Resp33", SqlDbType.Int));
sqlAlta.Parameters["@Resp33"].Value = _A33;
sqlAlta.Parameters.Add(new SqlParameter("@Resp34", SqlDbType.Int));
sqlAlta.Parameters["@Resp34"].Value = _A34;
sqlAlta.Parameters.Add(new SqlParameter("@Resp35", SqlDbType.Int));
sqlAlta.Parameters["@Resp35"].Value = _A35;
sqlAlta.Parameters.Add(new SqlParameter("@Resp36", SqlDbType.Int));
sqlAlta.Parameters["@Resp36"].Value = _A36;
sqlAlta.Parameters.Add(new SqlParameter("@Resp37", SqlDbType.Int));
```

```
sqlAlta.Parameters["@Resp37"].Value = _A37;
sqlAlta.Parameters.Add(new SqlParameter("@Resp38", SqlDbType.Int));
sqlAlta.Parameters["@Resp38"].Value = _A38;
sqlAlta.Parameters.Add(new SqlParameter("@Resp39", SqlDbType.Int));
sqlAlta.Parameters["@Resp39"].Value = _A39;
sqlAlta.Parameters.Add(new SqlParameter("@Resp40", SqlDbType.Int));
sqlAlta.Parameters["@Resp40"].Value = _A40;

sqlAlta.Parameters.Add(new SqlParameter("@Resp41", SqlDbType.Int));
sqlAlta.Parameters["@Resp41"].Value = _A41;
sqlAlta.Parameters.Add(new SqlParameter("@Resp42", SqlDbType.Int));
sqlAlta.Parameters["@Resp42"].Value = _A42;
sqlAlta.Parameters.Add(new SqlParameter("@Resp43", SqlDbType.Int));
sqlAlta.Parameters["@Resp43"].Value = _A43;
sqlAlta.Parameters.Add(new SqlParameter("@Resp44", SqlDbType.Int));
sqlAlta.Parameters["@Resp44"].Value = _A44;
sqlAlta.Parameters.Add(new SqlParameter("@Resp45", SqlDbType.Int));
sqlAlta.Parameters["@Resp45"].Value = _A45;
sqlAlta.Parameters.Add(new SqlParameter("@Resp46", SqlDbType.Int));
sqlAlta.Parameters["@Resp46"].Value = _A46;
sqlAlta.Parameters.Add(new SqlParameter("@Resp47", SqlDbType.Int));
sqlAlta.Parameters["@Resp47"].Value = _A47;
sqlAlta.Parameters.Add(new SqlParameter("@Resp48", SqlDbType.Int));
sqlAlta.Parameters["@Resp48"].Value = _A48;
sqlAlta.Parameters.Add(new SqlParameter("@Resp49", SqlDbType.Int));
sqlAlta.Parameters["@Resp49"].Value = _A49;
sqlAlta.Parameters.Add(new SqlParameter("@Resp50", SqlDbType.Int));
sqlAlta.Parameters["@Resp50"].Value = _A50;
```

```
sqlAlta.Parameters.Add(new SqlParameter("@Resp51", SqlDbType.Int));
sqlAlta.Parameters["@Resp51"].Value = _A51;
sqlAlta.Parameters.Add(new SqlParameter("@Resp52", SqlDbType.Int));
sqlAlta.Parameters["@Resp52"].Value = _A52;
sqlAlta.Parameters.Add(new SqlParameter("@Resp53", SqlDbType.Int));
sqlAlta.Parameters["@Resp53"].Value = _A53;
sqlAlta.Parameters.Add(new SqlParameter("@Resp54", SqlDbType.Int));
sqlAlta.Parameters["@Resp54"].Value = _A54;
sqlAlta.Parameters.Add(new SqlParameter("@Resp55", SqlDbType.Int));
sqlAlta.Parameters["@Resp55"].Value = _A55;
sqlAlta.Parameters.Add(new SqlParameter("@Resp56", SqlDbType.Int));
sqlAlta.Parameters["@Resp56"].Value = _A56;
sqlAlta.Parameters.Add(new SqlParameter("@Resp57", SqlDbType.Int));
sqlAlta.Parameters["@Resp57"].Value = _A57;
sqlAlta.Parameters.Add(new SqlParameter("@Resp58", SqlDbType.Int));
sqlAlta.Parameters["@Resp58"].Value = _A58;
sqlAlta.Parameters.Add(new SqlParameter("@Resp59", SqlDbType.Int));
sqlAlta.Parameters["@Resp59"].Value = _A59;
sqlAlta.Parameters.Add(new SqlParameter("@Resp60", SqlDbType.Int));
sqlAlta.Parameters["@Resp60"].Value = _A60;

sqlAlta.Parameters.Add(new SqlParameter("@Resp61", SqlDbType.Int));
sqlAlta.Parameters["@Resp61"].Value = _A61;
sqlAlta.Parameters.Add(new SqlParameter("@Resp62", SqlDbType.Int));
sqlAlta.Parameters["@Resp62"].Value = _A62;
sqlAlta.Parameters.Add(new SqlParameter("@Resp63", SqlDbType.Int));
sqlAlta.Parameters["@Resp63"].Value = _A63;
sqlAlta.Parameters.Add(new SqlParameter("@Resp64", SqlDbType.Int));
sqlAlta.Parameters["@Resp64"].Value = _A64;
```

```
sqlAlta.Parameters.Add(new SqlParameter("@Resp65", SqlDbType.Int));
sqlAlta.Parameters["@Resp65"].Value = _A65;
sqlAlta.Parameters.Add(new SqlParameter("@Resp66", SqlDbType.Int));
sqlAlta.Parameters["@Resp66"].Value = _A66;
sqlAlta.Parameters.Add(new SqlParameter("@Resp67", SqlDbType.Int));
sqlAlta.Parameters["@Resp67"].Value = _A67;
sqlAlta.Parameters.Add(new SqlParameter("@Resp68", SqlDbType.Int));
sqlAlta.Parameters["@Resp68"].Value = _A68;
sqlAlta.Parameters.Add(new SqlParameter("@Resp69", SqlDbType.Int));
sqlAlta.Parameters["@Resp69"].Value = _A69;
sqlAlta.Parameters.Add(new SqlParameter("@Resp70", SqlDbType.Int));
sqlAlta.Parameters["@Resp70"].Value = _A70;
sqlAlta.Parameters.Add(new SqlParameter("@Resp71", SqlDbType.Int));
sqlAlta.Parameters["@Resp71"].Value = _A71;
sqlAlta.Parameters.Add(new SqlParameter("@Resp72", SqlDbType.Int));
sqlAlta.Parameters["@Resp72"].Value = _A72;
sqlAlta.Parameters.Add(new SqlParameter("@Resp73", SqlDbType.Int));
sqlAlta.Parameters["@Resp73"].Value = _A73;
sqlAlta.Parameters.Add(new SqlParameter("@Resp74", SqlDbType.Int));
sqlAlta.Parameters["@Resp74"].Value = _A74;
sqlAlta.Parameters.Add(new SqlParameter("@Resp75", SqlDbType.Int));
sqlAlta.Parameters["@Resp75"].Value = _A75;
sqlAlta.Parameters.Add(new SqlParameter("@Resp76", SqlDbType.Int));
sqlAlta.Parameters["@Resp76"].Value = _A76;
sqlAlta.Parameters.Add(new SqlParameter("@Resp77", SqlDbType.Int));
sqlAlta.Parameters["@Resp77"].Value = _A77;
sqlAlta.Parameters.Add(new SqlParameter("@Resp78", SqlDbType.Int));
sqlAlta.Parameters["@Resp78"].Value = _A78;
sqlAlta.Parameters.Add(new SqlParameter("@Resp79", SqlDbType.Int));
```

```
sqlAlta.Parameters["@Resp79"].Value = _A79;
sqlAlta.Parameters.Add(new SqlParameter("@Resp80", SqlDbType.Int));
sqlAlta.Parameters["@Resp80"].Value = _A80;
sqlAlta.Parameters.Add(new SqlParameter("@RespFin", SqlDbType.NVarChar));
sqlAlta.Parameters["@RespFin"].Value = _Result;
//Cuestionario Motivacional
sqlAlta.Parameters.Add(new SqlParameter("@OrientMotR1", SqlDbType.Int));
sqlAlta.Parameters["@OrientMotR1"].Value = _OrientMot01;
sqlAlta.Parameters.Add(new SqlParameter("@OrientMotR2", SqlDbType.Int));
sqlAlta.Parameters["@OrientMotR2"].Value = _OrientMot02;
sqlAlta.Parameters.Add(new SqlParameter("@OrientMotR3", SqlDbType.Int));
sqlAlta.Parameters["@OrientMotR3"].Value = _OrientMot03;
sqlAlta.Parameters.Add(new SqlParameter("@OrientMotR4", SqlDbType.Int));
sqlAlta.Parameters["@OrientMotR4"].Value = _OrientMot04;
sqlAlta.Parameters.Add(new SqlParameter("@OrientMotR5", SqlDbType.Int));
sqlAlta.Parameters["@OrientMotR5"].Value = _OrientMot05;
sqlAlta.Parameters.Add(new SqlParameter("@OrientMotR6", SqlDbType.Int));
sqlAlta.Parameters["@OrientMotR6"].Value = _OrientMot06;
sqlAlta.Parameters.Add(new SqlParameter("@OrientMotR7", SqlDbType.Int));
sqlAlta.Parameters["@OrientMotR7"].Value = _OrientMot07;
sqlAlta.Parameters.Add(new SqlParameter("@OrientMotR8", SqlDbType.Int));
sqlAlta.Parameters["@OrientMotR8"].Value = _OrientMot08;
sqlAlta.Parameters.Add(new SqlParameter("@Resultado", SqlDbType.VarChar));
sqlAlta.Parameters["@Resultado"].Value = _Resultado;
try
{
    sqlAlta.ExecuteNonQuery();
    mObjConexion.Close();
    return 1;
```

```
        }
        catch
        {
            mObjConexion.Close();
            return 0;
        }

    }//fin alta registro
    //Lectura ID
    public int LecturaID()
    {
        int id;
        SqlDataReader objDataReader;
        string comando = "SELECT ID FROM Cuestionarios WHERE Cuestionarios.RespFin='"
+_Result + "'";
        SqlCommand objcomando = new SqlCommand(comando, mObjConexion);
        objDataReader = objcomando.ExecuteReader();
        objDataReader.Read();
        id = objDataReader.GetInt32(0);
        objDataReader.Close();
        return id;
    }//Lectura estilo de aprendizaje
    public string ObtenerNombre()
    {
        string cadena,cadena1,cadena2,cadena3;
        SqlDataReader objDataReader;
        string comando = "SELECT Nombre,ApellidoPaterno,ApellidoMaterno FROM
Cuestionarios WHERE Cuestionarios.ID='" + _ID + "'";
        SqlCommand objcomando = new SqlCommand(comando, mObjConexion);
        objDataReader = objcomando.ExecuteReader();
```

```
        objDataReader.Read();
        cadena1 = objDataReader.GetString(0);
        cadena2 = objDataReader.GetString(1);
        cadena3 = objDataReader.GetString(2);
        objDataReader.Close();
        cadena = cadena1 + " " + cadena2 + " " + cadena3;
        return cadena;
    }//Lectura estilo de aprendizaje
    public string EstiloAprendizaje()
    {
        string cadena;
        SqlDataReader objDataReader;
        string comando = "SELECT RespFin FROM Cuestionarios WHERE Cuestionarios.ID='" +
_ID + "'";
        SqlCommand objcomando = new SqlCommand(comando, mObjConexion);
        objDataReader = objcomando.ExecuteReader();
        objDataReader.Read();
        cadena = objDataReader.GetString(0);
        objDataReader.Close();
        return cadena;
    }//Lectura estilo de aprendizaje


    public int Chaea01()
    {
        string sqlActualiza = "UPDATE Cuestionarios SET Cuestionarios.Resp01='" + _A01 +
"',Cuestionarios.Resp02='" + _A02 + "',Cuestionarios.Resp03='" + _A03 +
"',Cuestionarios.Resp04='" + _A04 + "',Cuestionarios.Resp05='" + _A05 +
"',Cuestionarios.Resp06='" + _A06 + "',Cuestionarios.Resp07='" + _A07 +
"',Cuestionarios.Resp08='" + _A08 + "',Cuestionarios.RespFin='"+_Result+"' WHERE
Cuestionarios.ID='" + _ID + "'";
        SqlCommand objComando = new SqlCommand(sqlActualiza, mObjConexion);
        try
```

```
        {
            objComando.ExecuteNonQuery();
            mObjConexion.Close();
            return 1;
        }
        catch
        {
            mObjConexion.Close();
            return 0;
        }
    }
    public int Chaea02()
    {
        string sqlActualiza = "UPDATE Cuestionarios SET Cuestionarios.Resp09='" + _A09 +
"',Cuestionarios.Resp10='" + _A10+ "',Cuestionarios.Resp11='" + _A11 +
"',Cuestionarios.Resp12='" + _A12 + "',Cuestionarios.Resp13='" + _A13 +
"',Cuestionarios.Resp14='" + _A14 + "',Cuestionarios.Resp15='" + _A15 +
"',Cuestionarios.Resp16='" + _A16 + "' WHERE Cuestionarios.ID='" + _ID + "'";
        SqlCommand objComando = new SqlCommand(sqlActualiza, mObjConexion);
        try
        {
            objComando.ExecuteNonQuery();
            mObjConexion.Close();
            return 1;
        }
        catch
        {
            mObjConexion.Close();
            return 0;
        }
    }
```

```
    public int Chaea03()

    {

        string sqlActualiza = "UPDATE Cuestionarios SET Cuestionarios.Resp17='" + _A17 +
"',Cuestionarios.Resp18='"     +     _A18     +     "',Cuestionarios.Resp19='"     +     _A19     +
"',Cuestionarios.Resp20='"     +     _A20     +     "',Cuestionarios.Resp21='"     +     _A21     +
"',Cuestionarios.Resp22='"     +     _A22     +     "',Cuestionarios.Resp23='"     +     _A23     +
"',Cuestionarios.Resp24='" + _A24 + "' WHERE Cuestionarios.ID='" + _ID + "'";

        SqlCommand objComando = new SqlCommand(sqlActualiza, mObjConexion);

        try

        {

            objComando.ExecuteNonQuery();

            mObjConexion.Close();

            return 1;

        }

        catch

        {

            mObjConexion.Close();

            return 0;

        }

    }


    public int Chaea04()

    {

        string sqlActualiza = "UPDATE Cuestionarios SET Cuestionarios.Resp25='" + _A25 +
"',Cuestionarios.Resp26='"     +     _A26     +     "',Cuestionarios.Resp27='"     +     _A27     +
"',Cuestionarios.Resp28='"     +     _A28     +     "',Cuestionarios.Resp29='"     +     _A29     +
"',Cuestionarios.Resp30='"     +     _A30     +     "',Cuestionarios.Resp31='"     +     _A31     +
"',Cuestionarios.Resp32='" + _A32 + "' WHERE Cuestionarios.ID='" + _ID + "'";

        SqlCommand objComando = new SqlCommand(sqlActualiza, mObjConexion);

        try

        {

            objComando.ExecuteNonQuery();

            mObjConexion.Close();
```

```
            return 1;
        }
        catch
        {
            mObjConexion.Close();
            return 0;
        }
    }
    public int Chaea05()
    {
        string sqlActualiza = "UPDATE Cuestionarios SET Cuestionarios.Resp33='" + _A33 +
"',Cuestionarios.Resp34='" + _A34 + "',Cuestionarios.Resp35='" + _A35 +
"',Cuestionarios.Resp36='" + _A36 + "',Cuestionarios.Resp37='" + _A37 +
"',Cuestionarios.Resp38='" + _A38 + "',Cuestionarios.Resp39='" + _A39 +
"',Cuestionarios.Resp40='" + _A40 + "' WHERE Cuestionarios.ID='" + _ID + "'";
        SqlCommand objComando = new SqlCommand(sqlActualiza, mObjConexion);
        try
        {
            objComando.ExecuteNonQuery();
            mObjConexion.Close();
            return 1;
        }
        catch
        {
            mObjConexion.Close();
            return 0;
        }
    }

    public int Chaea06()
    {
```

```
        string sqlActualiza = "UPDATE Cuestionarios SET Cuestionarios.Resp41='" + _A41 +
'',Cuestionarios.Resp42='' + _A42 + '',Cuestionarios.Resp43='' + _A43 +
'',Cuestionarios.Resp44='' + _A44 + '',Cuestionarios.Resp45='' + _A45 +
'',Cuestionarios.Resp46='' + _A46 + '',Cuestionarios.Resp47='' + _A47 +
'',Cuestionarios.Resp48='' + _A48 + '' WHERE Cuestionarios.ID='' + _ID + '''';

        SqlCommand objComando = new SqlCommand(sqlActualiza, mObjConexion);

        try

        {

            objComando.ExecuteNonQuery();

            mObjConexion.Close();

            return 1;

        }

        catch

        {

            mObjConexion.Close();

            return 0;

        }

    }


    public int Chaea07()

    {

        string sqlActualiza = "UPDATE Cuestionarios SET Cuestionarios.Resp49='" + _A49 +
'',Cuestionarios.Resp50='' + _A50 + '',Cuestionarios.Resp51='' + _A51 +
'',Cuestionarios.Resp52='' + _A52 + '',Cuestionarios.Resp53='' + _A53 +
'',Cuestionarios.Resp54='' + _A54 + '',Cuestionarios.Resp55='' + _A55 +
'',Cuestionarios.Resp56='' + _A56 + '' WHERE Cuestionarios.ID='' + _ID + '''';

        SqlCommand objComando = new SqlCommand(sqlActualiza, mObjConexion);

        try

        {

            objComando.ExecuteNonQuery();

            mObjConexion.Close();

            return 1;

        }
```

```
        catch
        {
            mObjConexion.Close();
            return 0;
         }
    }
    public int Chaea08()
    {
        string sqlActualiza = "UPDATE Cuestionarios SET Cuestionarios.Resp57='" + _A57 +
"',Cuestionarios.Resp58='" + _A58 + "',Cuestionarios.Resp59='" + _A59 +
"',Cuestionarios.Resp60='" + _A60 + "',Cuestionarios.Resp61='" + _A61 +
"',Cuestionarios.Resp62='" + _A62 + "',Cuestionarios.Resp63='" + _A63 +
"',Cuestionarios.Resp64='" + _A64 + "' WHERE Cuestionarios.ID='" + _ID + "'";
        SqlCommand objComando = new SqlCommand(sqlActualiza, mObjConexion);
        try
        {
            objComando.ExecuteNonQuery();
            mObjConexion.Close();
            return 1;
        }
        catch
        {
            mObjConexion.Close();
            return 0;
        }
    }
    public int Chaea09()
    {
        string sqlActualiza = "UPDATE Cuestionarios SET Cuestionarios.Resp65='" + _A65 +
"',Cuestionarios.Resp66='" + _A66 + "',Cuestionarios.Resp67='" + _A67 +
"',Cuestionarios.Resp68='" + _A68 + "',Cuestionarios.Resp69='" + _A69 +
"',Cuestionarios.Resp70='" + _A70 + "',Cuestionarios.Resp71='" + _A71 +
"',Cuestionarios.Resp72='" + _A72 + "' WHERE Cuestionarios.ID='" + _ID + "'";
```

```
        SqlCommand objComando = new SqlCommand(sqlActualiza, mObjConexion);
        try
        {
            objComando.ExecuteNonQuery();
            mObjConexion.Close();
            return 1;
        }
        catch
        {
            mObjConexion.Close();
            return 0;
        }
    }
    public int Chaea10()
    {
        string sqlActualiza = "UPDATE Cuestionarios SET Cuestionarios.Resp73='" + _A73 +
"',Cuestionarios.Resp74='" + _A74 + "',Cuestionarios.Resp75='" + _A75 +
"',Cuestionarios.Resp76='" + _A76 + "',Cuestionarios.Resp77='" + _A77 +
"',Cuestionarios.Resp78='" + _A78 + "',Cuestionarios.Resp79='" + _A79 +
"',Cuestionarios.Resp80='" + _A80 + "' WHERE Cuestionarios.ID='" + _ID + "'";
        SqlCommand objComando = new SqlCommand(sqlActualiza, mObjConexion);
        try
        {
            objComando.ExecuteNonQuery();
            mObjConexion.Close();
            return 1;
        }
        catch
        {
            mObjConexion.Close();
            return 0;
```

```
        }
    }
    //analisis para el tipo de aprendizaje obtenido
    public string TipoAprendizaje()
    {
        int activo = 0, reflexivo = 0, teorico = 0, pragmatico = 0, grande = 0;
        string mayor;
        SqlDataReader objDataReader;
        string sqlConsulta = "SELECT * FROM Cuestionarios WHERE Cuestionarios.ID='" + _ID +
"'";
        SqlCommand consulta = new SqlCommand(sqlConsulta, mObjConexion);
        objDataReader = consulta.ExecuteReader();
        while (objDataReader.Read())
        {
            _Nombre = objDataReader.GetString(1);
            _ApellidoPaterno=objDataReader.GetString(2);
            _ApellidoMaterno = objDataReader.GetString(3);
            _Sexo = objDataReader.GetString(4);
            _Nacionalidad = objDataReader.GetString(5);
            _FechaNac = objDataReader.GetString(6);
            _Matricula = objDataReader.GetString(7);
            _Carrera = objDataReader.GetString(8);

            _A01 = objDataReader.GetInt32(9);
            _A02 = objDataReader.GetInt32(10);
            _A03 = objDataReader.GetInt32(11);
            _A04 = objDataReader.GetInt32(12);
            _A05 = objDataReader.GetInt32(13);
            _A06 = objDataReader.GetInt32(14);
            _A07 = objDataReader.GetInt32(15);
```

```
_A08 = objDataReader.GetInt32(16);
_A09 = objDataReader.GetInt32(17);
_A10 = objDataReader.GetInt32(18);
_A11 = objDataReader.GetInt32(19);
_A12 = objDataReader.GetInt32(20);
_A13 = objDataReader.GetInt32(21);
_A14 = objDataReader.GetInt32(22);
_A15 = objDataReader.GetInt32(23);
_A16 = objDataReader.GetInt32(24);
_A17 = objDataReader.GetInt32(25);
_A18 = objDataReader.GetInt32(26);
_A19 = objDataReader.GetInt32(27);
_A20 = objDataReader.GetInt32(28);
_A21 = objDataReader.GetInt32(29);
_A22 = objDataReader.GetInt32(30);
_A23 = objDataReader.GetInt32(31);
_A24 = objDataReader.GetInt32(32);
_A25 = objDataReader.GetInt32(33);
_A26 = objDataReader.GetInt32(34);
_A27 = objDataReader.GetInt32(35);
_A28 = objDataReader.GetInt32(36);
_A29 = objDataReader.GetInt32(37);
_A30 = objDataReader.GetInt32(38);
_A31 = objDataReader.GetInt32(39);
_A32 = objDataReader.GetInt32(40);
_A33 = objDataReader.GetInt32(41);
_A34 = objDataReader.GetInt32(42);
_A35 = objDataReader.GetInt32(43);
_A36 = objDataReader.GetInt32(44);
```

```
_A37 = objDataReader.GetInt32(45);
_A38 = objDataReader.GetInt32(46);
_A39 = objDataReader.GetInt32(47);
_A40 = objDataReader.GetInt32(48);
_A41 = objDataReader.GetInt32(49);
_A42 = objDataReader.GetInt32(50);
_A43 = objDataReader.GetInt32(51);
_A44 = objDataReader.GetInt32(52);
_A45 = objDataReader.GetInt32(53);
_A46 = objDataReader.GetInt32(54);
_A47 = objDataReader.GetInt32(55);
_A48 = objDataReader.GetInt32(56);
_A49 = objDataReader.GetInt32(57);
_A50 = objDataReader.GetInt32(58);
_A51 = objDataReader.GetInt32(59);
_A52 = objDataReader.GetInt32(60);
_A53 = objDataReader.GetInt32(61);
_A54 = objDataReader.GetInt32(62);
_A55 = objDataReader.GetInt32(63);
_A56 = objDataReader.GetInt32(64);
_A57 = objDataReader.GetInt32(65);
_A58 = objDataReader.GetInt32(66);
_A59 = objDataReader.GetInt32(67);
_A60 = objDataReader.GetInt32(68);
_A61 = objDataReader.GetInt32(69);
_A62 = objDataReader.GetInt32(70);
_A63 = objDataReader.GetInt32(71);
_A64 = objDataReader.GetInt32(72);
_A65 = objDataReader.GetInt32(73);
```

```
        _A66 = objDataReader.GetInt32(74);
        _A67 = objDataReader.GetInt32(75);
        _A68 = objDataReader.GetInt32(76);
        _A69 = objDataReader.GetInt32(77);
        _A70 = objDataReader.GetInt32(78);
        _A71 = objDataReader.GetInt32(79);
        _A72 = objDataReader.GetInt32(80);
        _A73 = objDataReader.GetInt32(81);
        _A74 = objDataReader.GetInt32(82);
        _A75 = objDataReader.GetInt32(83);
        _A76 = objDataReader.GetInt32(84);
        _A77 = objDataReader.GetInt32(85);
        _A78 = objDataReader.GetInt32(86);
        _A79 = objDataReader.GetInt32(87);
        _A80 = objDataReader.GetInt32(88);
    }
    if (_A01 == 1)
        pragmatico++;
    if (_A02 == 1)
        teorico++;
    if (_A03 == 1)
        activo++;
    if (_A04 == 1)
        teorico++;
    if (_A05 == 1)
        activo++;
    if (_A06 == 1)
        teorico++;
    if (_A07 == 1)
```

```
    activo++;
if (_A08 == 1)
    pragmatico++;
if (_A09 == 1)
    activo++;
if (_A10 == 1)
    reflexivo++;
if (_A11 == 1)
    teorico++;
if (_A12 == 1)
    pragmatico++;
if (_A13 == 1)
    activo++;
if (_A14 == 1)
    pragmatico++;
if (_A15 == 1)
    teorico++;
if (_A16 == 1)
    reflexivo++;
if (_A17 == 1)
    teorico++;
if (_A18 == 1)
    reflexivo++;
if (_A19 == 1)
    reflexivo++;
if (_A20 == 1)
    activo++;
if (_A21 == 1)
    teorico++;
```

```
if (_A22 == 1)
    pragmatico++;
if (_A23 == 1)
    teorico++;
if (_A24 == 1)
    pragmatico++;
if (_A25 == 1)
    teorico++;
if (_A26 == 1)
    activo++;
if (_A27 == 1)
    activo++;
if (_A28 == 1)
    reflexivo++;
if (_A29 == 1)
    teorico++;
if (_A30 == 1)
    pragmatico++;
if (_A31 == 1)
    reflexivo++;
if (_A32 == 1)
    reflexivo++;
if (_A33 == 1)
    teorico++;
if (_A34 == 1)
    reflexivo++;
if (_A35 == 1)
    activo++;
if (_A36 == 1)
```

```
    reflexivo++;
if (_A37 == 1)
    activo++;
if (_A38 == 1)
    pragmatico++;
if (_A39 == 1)
    reflexivo++;
if (_A40 == 1)
    pragmatico++;
if (_A41 == 1)
    activo++;
if (_A42 == 1)
    reflexivo++;
if (_A43 == 1)
    activo++;
if (_A44 == 1)
    reflexivo++;
if (_A45 == 1)
    teorico++;
if (_A46 == 1)
    activo++;
if (_A47 == 1)
    pragmatico++;
if (_A48 == 1)
    activo++;
if (_A49 == 1)
    reflexivo++;
if (_A50 == 1)
    teorico++;
```

```
if (_A51 == 1)
    activo++;
if (_A52 == 1)
    pragmatico++;
if (_A53 == 1)
    pragmatico++;
if (_A54 == 1)
    teorico++;
if (_A55 == 1)
    reflexivo++;
if (_A56 == 1)
    pragmatico++;
if (_A57 == 1)
    pragmatico++;
if (_A58 == 1)
    reflexivo++;
if (_A59 == 1)
    pragmatico++;
if (_A60 == 1)
    teorico++;
if (_A61 == 1)
    activo++;
if (_A62 == 1)
    pragmatico++;
if (_A63 == 1)
    reflexivo++;
if (_A64 == 1)
    teorico++;
if (_A65 == 1)
```

```
    reflexivo++;
if (_A66 == 1)
    teorico++;
if (_A67 == 1)
    activo++;
if (_A68 == 1)
    pragmatico++;
if (_A69 == 1)
    reflexivo++;
if (_A70 == 1)
    reflexivo++;
if (_A71 == 1)
    teorico++;
if (_A72 == 1)
    pragmatico++;
if (_A73 == 1)
    pragmatico++;
if (_A74 == 1)
    activo++;
if (_A75 == 1)
    activo++;
if (_A76 == 1)
    pragmatico++;
if (_A77 == 1)
    activo++;
if (_A78 == 1)
    teorico++;
if (_A79 == 1)
    reflexivo++;
```

```
    if (_A80 == 1)
        teorico++;
    if (activo > reflexivo)
    {
        grande = activo;
        mayor = "Activo";
    }
    else
    {
        grande = reflexivo;
        mayor = "Reflexivo";
    }

    if (grande < teorico)
    {
        grande = teorico;
        mayor = "Teorico";
    }

    if (grande < pragmatico)
    {
        grande = pragmatico;
        mayor = "Pragmatico";
    }
    objDataReader.Close();
    return mayor;
}
public void FinChaea()
{
```

```
        string sqlresultado = "UPDATE Cuestionarios SET Cuestionarios.RespFin='" + _Result + "'
WHERE Cuestionarios.ID='" + _ID + "'";

        SqlCommand comando = new SqlCommand(sqlresultado, mObjConexion);

        comando.ExecuteNonQuery();

        mObjConexion.Close();

    }

    //Cuestionario 2

    public int Update01()

    {

        string sqlactualizar = "UPDATE Cuestionarios SET Cuestionarios.OrientMotR1='" +
_OrientMot01 + "',Cuestionarios.OrientMotR2='" + _OrientMot02 +
"',Cuestionarios.OrientMotR3='" + _OrientMot03 + "'WHERE Cuestionarios.ID='" + _ID + "'";

        SqlCommand objcomando = new SqlCommand(sqlactualizar, mObjConexion);


        try

        {

            objcomando.ExecuteNonQuery();

            mObjConexion.Close();

            return 1;

        }

        catch

        {

            mObjConexion.Close();

            return 0;

        }

    }


    public int Update02()

    {

        string sqlactualizar = "UPDATE Cuestionarios SET Cuestionarios.OrientMotR4='" +
_OrientMot04 + "',Cuestionarios.OrientMotR5='" + _OrientMot05 +
"',Cuestionarios.OrientMotR6='" + _OrientMot06 + "' WHERE Cuestionarios.ID='" + _ID + "'";
```

```
        SqlCommand objcomando = new SqlCommand(sqlactualizar, mObjConexion);
        try
        {
            objcomando.ExecuteNonQuery();
            mObjConexion.Close();
            return 1;
        }
        catch
        {
            mObjConexion.Close();
            return 0;
        }
    }
    public int Update03()
    {
        string sqlactualizar = "UPDATE Cuestionarios SET Cuestionarios.OrientMotR7='" +
_OrientMot07 + "',Cuestionarios.OrientMotR8='" + _OrientMot08 + "',Cuestionarios.Resultado='"
+ _Resultado + "' WHERE Cuestionarios.ID='" + _ID + "'";
        SqlCommand objcomando = new SqlCommand(sqlactualizar, mObjConexion);
        try
        {
            objcomando.ExecuteNonQuery();
            mObjConexion.Close();
            return 1;
        }
        catch
        {
            mObjConexion.Close();
            return 0;
        }
```

```
    }


    public string ObjetivoAprendizaje()
    {
        int MasterGoal, PerformaceGoal;
        SqlDataReader objDataReader;
        string sqlConsulta = "SELECT OrientMotR1,OrientMotR2,OrientMotR3,OrientMotR4,OrientMotR5,OrientMotR6,OrientMotR7,OrientMotR8 FROM Cuestionarios WHERE Cuestionarios.ID='" + _ID + "'";
        SqlCommand consulta = new SqlCommand(sqlConsulta, mObjConexion);
        objDataReader = consulta.ExecuteReader();
        while (objDataReader.Read())
        {
            _OrientMot01 = objDataReader.GetInt32(0);
            _OrientMot02 = objDataReader.GetInt32(1);
            _OrientMot03 = objDataReader.GetInt32(2);
            _OrientMot04 = objDataReader.GetInt32(3);
            _OrientMot05 = objDataReader.GetInt32(4);
            _OrientMot06 = objDataReader.GetInt32(5);
            _OrientMot07 = objDataReader.GetInt32(6);
            _OrientMot08 = objDataReader.GetInt32(7);
        }
        MasterGoal = _OrientMot01 + _OrientMot02 + _OrientMot03 + _OrientMot04;
        PerformaceGoal = _OrientMot05 + _OrientMot06 + _OrientMot07 + _OrientMot08;
        if (MasterGoal == PerformaceGoal)
            {
                objDataReader.Close();
                return "Internos y Externo";
            }
```

```
            if (MasterGoal < PerformaceGoal)
            {
                objDataReader.Close();
                return "Externos";
            }
            else
            {
                objDataReader.Close();
                return "Internos";
            }
    }
    public void FinOrientacion()
    {
        string sqlresultado = "UPDATE Cuestionarios SET Cuestionarios.Resultado='" +
_Resultado + "' WHERE Cuestionarios.ID='" + _ID + "'";
        SqlCommand comando = new SqlCommand(sqlresultado, mObjConexion);
        comando.ExecuteNonQuery();
        mObjConexion.Close();

    }
}//fin de la Clase Cuestionarioip
```

En la primera página del cuestionario tenemos las primeras ocho preguntas (de la pregunta 1 a la pregunta 8), como se muestra en la Figura 4.

Figura A3.5 Pagina01.aspx

El código de diseño de la pagina (Pagina01.aspx), solamente contiene tablas con las preguntas.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Chaea01.aspx.cs" Inherits="Chaea01" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head id="Head1" runat="server">

   <title>Chaea</title>

   <style type="text/css">

<!--

@import url("tipografs sitio.css");

body {

        margin-left: 0px;

        margin-top: 0px;

        margin-right: 0px;

        margin-bottom: 0px;

}

-->

</style>
```

```
<script type="text/JavaScript">
<!--
function MM_swapImgRestore() { //v3.0
  var i,x,a=document.MM_sr; for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;
}
function MM_preloadImages() { //v3.0
  var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
    var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)
    if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}
function MM_findObj(n, d) { //v4.01
  var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}
function MM_swapImage() { //v3.0
  var  i,j=0,x,a=MM_swapImage.arguments;  document.MM_sr=new  Array;  for(i=0;i<(a.length-
2);i+=3)
   if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!x.oSrc) x.oSrc=x.src; x.src=a[i+2];}
}
//-->
</script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <table width="1000" height="650" border="0" align="center" cellpadding="0" cellspacing="0"
bgcolor="#F2E79F">
```

```
 <tr>

  <td width="250" height="90" bgcolor="#A39994" class="CuestTitulos">

  <div align="right">

    <span style="font-size: 12pt">CUESTIONARIO HONEY-ALONSO DE ESTILOS DE APRENDIZAJE </span>

  </div></td>

  <td width="230" height="90"><img src="Imagenes/C-cuest1.gif" width="230" height="90" alt="copete" /></td>

  <td width="285" height="90"></td>

  <td height="90" align="center" valign="bottom" style="width: 185px"><img src="Imagenes/logo-gris.png" width="185" height="49" alt="logos" /></td>

  <td width="50" height="90"></td>

 </tr>

 <tr>

  <td height="50" colspan="5"></td>

 </tr>

 <tr>

  <td height="410" colspan="5" style="text-align: center">

    <strong><span style="font-family: Arial">INSTRUCCIONES PARA RESPONDER AL CUESTIONARIO

    </span>

      <br />

    </strong>

    <ul>

      <li style="text-align: center"><span style="font-family: Arial">Este cuestionario ha sido diseñado para identificar

        su Estilo preferido de Aprendizaje</span><br />

      </li>

      <li style="text-align: center"><span style="font-family: Arial">

        No es un test de Inteligencia, ni de personalidad..</span><br />

      </li>
```

<li style="text-align: center">No hay tiempo límite de tiempo para constestar

al Cuestionario.

No le ocupará más de   15 minutos.

<li style="text-align: center">No hay respuestas correctas o erróneas.Será útil

en la medida que sea sincero/a en

sus respuestas.

<li style="text-align: center">Si está más de acuerdo que en desacuerdo con el

ítem seleccione el signo (+),

si por el contrario, está más en desacuerdo seleccione el signo (-).

<li style="text-align: center">Por favor conteste todos los items.

<li style="text-align: center">El cuestionario es anónimo. Para facilitar el análisis

del grupo le rogamos que

responda también a las preguntas de índole socioacadémica.

Muchas gracias!!!!

```
        <p>
            <br />
        </p>
    </td>
  </tr>
  <tr>
    <td height="50" colspan="3"><table width="500" border="0" align="right" cellpadding="0"
cellspacing="0">
  <tr>
    <td align="center" style="height: 18px">
        <asp:Label ID="LblID" runat="server" Visible="False"></asp:Label></td>
  </tr>
</table>
</td>
    <td height="50" align="right" valign="middle" style="width: 185px">
        <asp:Button ID="CmdSiguiente" runat="server" BackColor="SeaGreen" Font-Bold="True"
            Font-Size="X-Large" ForeColor="White" Height="37px" Text="→"
OnClick="CmdSiguiente_Click" />
            </td>
    <td width="50" height="50"></td>
  </tr>
  <tr>
    <td height="50" colspan="5"></td>
  </tr>
</table>


    </div></form>
</body>
</html>
```

Código de Programación

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class Chaea01 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {

            if (Request.Cookies["userID"] != null)
            {
                LblID.Text = Server.HtmlEncode(Request.Cookies["userID"].Value);
            }
            else
                Server.Transfer("Default.aspx");
        }
    }
    protected void CmdSiguiente_Click(object sender, EventArgs e)
    {
```

```
        Server.Transfer("Chaea02.aspx");
    }
}
```

En las páginas Pagina02.aspx, Pagina03.aspx, Pagina04.aspx,…, Pagina10.aspx, el código de programación es el mismo solo cambia la captura de los datos.

El resultado de la evaluación del cuestionario se muestra por medio de la página Resultado.aspx, como se muestra en la Figura 5.

Figura A3.6 Pantalla de Resultado del Cuestionario CHAEA.

Código de Programación Resultado.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class Chaea12 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {

            if (Request.Cookies["userID"] != null)
            {
                LblID.Text = Server.HtmlEncode(Request.Cookies["userID"].Value);
            }
            else
                Server.Transfer("Default.aspx");
        }
        Cuestionarioip final = new Cuestionarioip();
        int id = Convert.ToInt32(LblID.Text);
        final.ID = id;
        LblTipoAprend.Text = final.TipoAprendizaje();
        Cuestionarioip resultado = new Cuestionarioip();
        resultado.ID = id;
        resultado.Result = LblTipoAprend.Text;
        resultado.FinChaea();
    }
    protected void CmdTest2_Click(object sender, EventArgs e)
    {
        Server.Transfer("Orientacion01.aspx");
    }
```

Respaldo de los Datos del Estudiante en la Base de Datos

La base de datos a la cual se conecta la aplicación Web se llama **CHAEA** y es de SQL Server, contiene una Tabla llamada **Cuestionarios.**

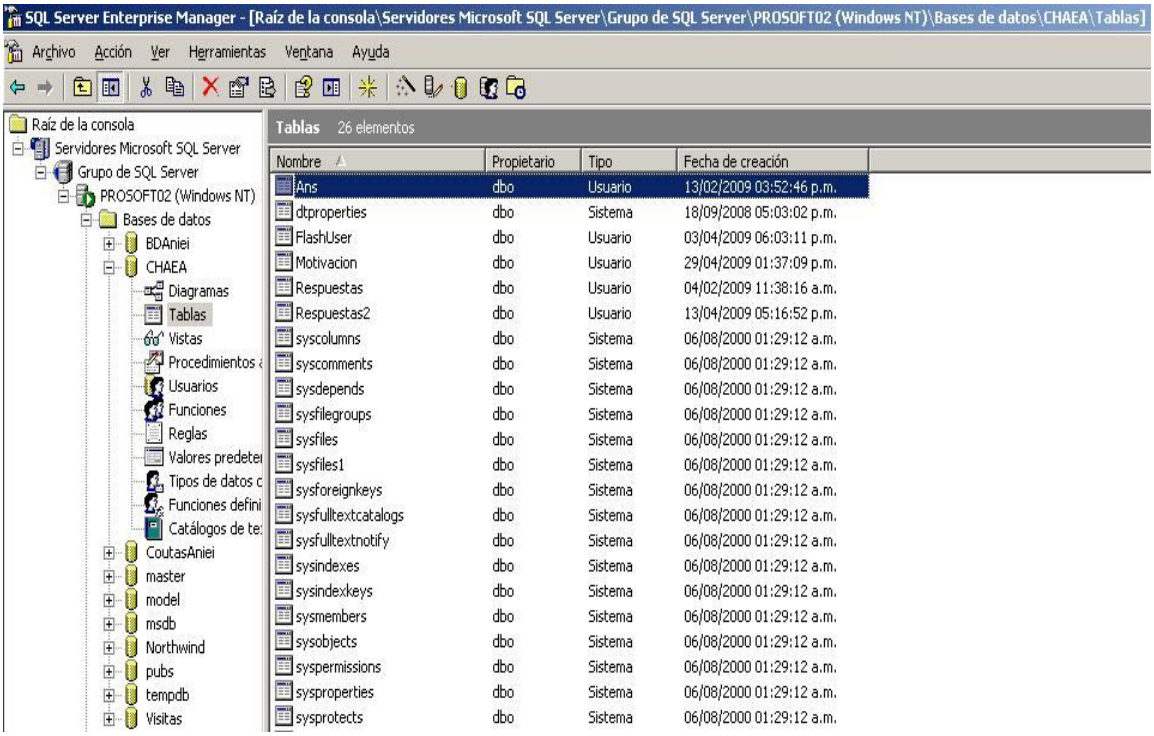


Figura A3.7 Pantalla de Presentación de la Estructura de la Base de Datos del Sistema ProgEst

La tabla **Cuestionario** tiene la siguiente estructura:

Nombre Columna	Tipo de Valor	
ID	int	Identificador del Registro
Resp01	int	Respuesta 1
Resp02	int	Respuesta 2
Resp03	int	Respuesta 3
Resp04	int	Respuesta 4
Resp05	int	Respuesta 5
Resp06	int	Respuesta 6
Resp07	int	Respuesta 7
Resp08	int	Respuesta 8
Resp09	int	Respuesta 9
Resp10	int	Respuesta 10
Resp11	int	Respuesta 11
Resp12	int	Respuesta 12
Resp13	int	Respuesta 13
Resp14	int	Respuesta 14
Resp15	int	Respuesta 15
Resp16	int	Respuesta 16
Resp17	int	Respuesta 17
Resp18	int	Respuesta 18
Resp19	int	Respuesta 19
Resp20	int	Respuesta 20
Resp21	int	Respuesta 21
Resp22	int	Respuesta 22
Resp23	int	Respuesta 23
Resp24	int	Respuesta 24
Resp25	int	Respuesta 25
Resp26	int	Respuesta 26
Resp27	int	Respuesta 27

Resp28	int	Respuesta 28
Resp29	int	Respuesta 29
Resp30	int	Respuesta 30
Resp31	int	Respuesta 31
Resp32	int	Respuesta 32
Resp33	int	Respuesta 33
Resp34	int	Respuesta 34
Resp35	int	Respuesta 35
Resp36	int	Respuesta 36
Resp37	int	Respuesta 37
Resp38	int	Respuesta 38
Resp39	int	Respuesta 39
Resp40	int	Respuesta 40
Resp41	int	Respuesta 41
Resp42	int	Respuesta 42
Resp43	int	Respuesta 43
Resp44	int	Respuesta 44
Resp45	int	Respuesta 45
Resp46	int	Respuesta 46
Resp47	int	Respuesta 47
Resp48	int	Respuesta 48
Resp49	int	Respuesta 49
Resp50	int	Respuesta 50
Resp51	int	Respuesta 51
Resp52	int	Respuesta 52
Resp53	int	Respuesta 53
Resp54	int	Respuesta 54
Resp55	int	Respuesta 55
Resp56	int	Respuesta 56
Resp57	int	Respuesta 57

Resp58	int	Respuesta 58
Resp59	int	Respuesta 59
Resp60	int	Respuesta 60
Resp61	int	Respuesta 61
Resp62	int	Respuesta 62
Resp63	int	Respuesta 63
Resp64	int	Respuesta 64
Resp65	int	Respuesta 65
Resp66	int	Respuesta 66
Resp67	int	Respuesta 67
Resp68	int	Respuesta 68
Resp69	int	Respuesta 69
Resp70	int	Respuesta 70
Resp71	int	Respuesta 71
Resp72	int	Respuesta 72
Resp73	int	Respuesta 73
Resp74	int	Respuesta 74
Resp75	int	Respuesta 75
Resp76	int	Respuesta 76
Resp77	int	Respuesta 77
Resp78	int	Respuesta 78
Resp79	int	Respuesta 79
Resp80	int	Respuesta 80
RespFin	Nvarchar	Tipo de Aprendizaje obtenido

A3.3 Cuestionario Orientación Motivación de Estudio

Figura A3.8 Pantalla de Resultado del Cuestionario de Motivación de Estudio.

Código de Diseño del Cuestionario de Motivación de Estudio

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Orientacion04.aspx.cs" Inherits="Orientacion04" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head id="Head1" runat="server">

   <title>Orientacion04</title>

 <style type="text/css">

<!--

@import url("tipografs sitio.css");

body {

        margin-left: 0px;

        margin-top: 0px;
```

```
            margin-right: 0px;
            margin-bottom: 0px;
}
-->
</style>
<script type="text/JavaScript">
<!--
function MM_swapImgRestore() { //v3.0
  var i,x,a=document.MM_sr; for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;
}

function MM_preloadImages() { //v3.0
  var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
    var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)
    if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}

function MM_findObj(n, d) { //v4.01
  var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}
function MM_swapImage() { //v3.0
  var  i,j=0,x,a=MM_swapImage.arguments;  document.MM_sr=new  Array;  for(i=0;i<(a.length-
2);i+=3)
   if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!x.oSrc) x.oSrc=x.src; x.src=a[i+2];}
}
//-->
```

```
</script>

</head>

<body >

<form id="form1" runat="server">

<div>

<table width="1000" height="650" border="0" align="center" cellpadding="0" cellspacing="0" bgcolor="#F2E79F">

  <tr>

    <td width="250" height="90" bgcolor="#A39994" class="CuestTitulos"><div align="right">

        <span style="font-size: 12pt">Cuestionario para determinar la motivación de estudio

          de los estudiantes</span></div></td>

    <td width="230" height="90"><img src="Imagenes/C-cuest1.gif" width="230" height="90" alt="copete" /></td>

    <td width="285" height="90"></td>

    <td height="90" align="center" valign="bottom" style="width: 187px"><img src="Imagenes/logo-gris.png" width="185" height="49" alt="logos" /></td>

    <td width="50" height="90"></td>

  </tr>

  <tr>

    <td height="50" colspan="5"></td>

  </tr>

  <tr>

    <td colspan="5" style="height: 410px" align="center">

        <p align="center" class="CuestMensajeBIG">

          Gracias <asp:Label ID="LbNombre" runat="server"></asp:Label>

        </p>

        <p align="center" class="CuestMensajeBIG">

          por completar la información</p>

        <p align="center" class="CuestMensajeBIG">

        </p>

        <p align="center" class="CuestMensajeBIG">
```

```
        </p>
        <p align="center" class="CuestMensaje">
            Espera un momento para cargar tu configuración</p>
        <asp:Label ID="LbObjetivo" runat="server" ForeColor="Maroon" Font-Names="Arial" Font-
    Size="Medium"></asp:Label></td>
  </tr>
  <tr>
    <td colspan="3" style="height: 50px"><table width="500" border="0" align="right"
    cellpadding="0" cellspacing="0">
  <tr>
  </tr>
</table>
        <asp:Label ID="LblID" runat="server" Visible="False"></asp:Label></td>
    <td align="right" valign="middle" style="width: 187px; height: 50px;">
        <asp:Button ID="CmdSiguiente" runat="server" BackColor="SeaGreen" Font-Bold="True"
            Font-Size="X-Large" ForeColor="White" Height="37px"
            Text="→" OnClick="CmdSiguiente_Click" /> </td>
    <td width="50" style="height: 50px"></td>
  </tr>
  <tr>
    <td colspan="5" style="height: 50px"></td>
  </tr>
</table>
</div></form>
</body>
</html>
```

Código de Programación del Cuestionario de Motivación de Estudio

```
using System;
```

```
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class Orientacion04 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string resultado;

        if (!IsPostBack)
        {
            if (Request.Cookies["userID"] != null)
            {
                LblID.Text = Server.HtmlEncode(Request.Cookies["userID"].Value);
            }
            else
                Server.Transfer("Default.aspx");
        }
       Cuestionarioip nom = new Cuestionarioip();
        nom.ID = Convert.ToInt32(LblID.Text);
        LbNombre.Text = nom.ObtenerNombre();
        //obtenemos el objetivo de aprendizaje
```

```
            Cuestionarioip objetivo = new Cuestionarioip();
            objetivo.ID = Convert.ToInt32(LblID.Text);
            resultado = objetivo.ObjetivoAprendizaje();
            LbObjetivo.Text="Motivo de aprendizaje: " + resultado;

            Cuestionarioip guardar = new Cuestionarioip();
            guardar.ID = Convert.ToInt32(LblID.Text);
            guardar.Resultado = resultado;
            guardar.FinOrientacion();
        }
        protected void CmdSiguiente_Click(object sender, EventArgs e)
        {
            Response.Redirect("Escenario.aspx");
        }
}
```

La tabla **Cuestionario** tiene la siguiente estructura:

Nombre Columna	Tipo de Valor	
ID	int	identificador del Registro
Resp01	int	Respuesta 1
Resp02	int	Respuesta 2
Resp03	int	Respuesta 3
Resp04	int	Respuesta 4
Resp05	int	Respuesta 5
Resp06	int	Respuesta 6
Resp07	int	Respuesta 7
Resp08	int	Respuesta 8

Resp09	int	Respuesta 9
Resp10	int	Respuesta 10
Resp11	int	Respuesta 11
Resp12	int	Respuesta 12
Resp13	int	Respuesta 13
Resp14	int	Respuesta 14
Resp15	int	Respuesta 15
Resp16	int	Respuesta 16
Resp17	int	Respuesta 17
Resp18	int	Respuesta 18
Resp19	int	Respuesta 19
Resp20	int	Respuesta 20
Resp21	int	Respuesta 21
Resp22	int	Respuesta 22
Resp23	int	Respuesta 23
Resp24	int	Respuesta 24
Resp25	int	Respuesta 25
Resp26	int	Respuesta 26
Resp27	int	Respuesta 27
Resp28	int	Respuesta 28
Resp29	int	Respuesta 29
Resp30	int	Respuesta 30
Resp31	int	Respuesta 31
Resp32	int	Respuesta 32
Resp33	int	Respuesta 33
Resp34	int	Respuesta 34
Resp35	int	Respuesta 35
Resp36	int	Respuesta 36

Resp37	int	Respuesta 37
Resp38	int	Respuesta 38
Resp39	int	Respuesta 39
Resp40	int	Respuesta 40
Resp41	int	Respuesta 41
Resp42	int	Respuesta 42
Resp43	int	Respuesta 43
Resp44	int	Respuesta 44
Resp45	int	Respuesta 45
Resp46	int	Respuesta 46
Resp47	int	Respuesta 47
Resp48	int	Respuesta 48
Resp49	int	Respuesta 49
Resp50	int	Respuesta 50
Resp51	int	Respuesta 51
Resp52	int	Respuesta 52
Resp53	int	Respuesta 53
Resp54	int	Respuesta 54
Resp55	int	Respuesta 55
Resp56	int	Respuesta 56
Resp57	int	Respuesta 57
Resp58	int	Respuesta 58
Resp59	int	Respuesta 59
Resp60	int	Respuesta 60
Resp61	int	Respuesta 61
Resp62	int	Respuesta 62
Resp63	int	Respuesta 63
Resp64	int	Respuesta 64

Resp65	int	Respuesta 65
Resp66	int	Respuesta 66
Resp67	int	Respuesta 67
Resp68	int	Respuesta 68
Resp69	int	Respuesta 69
Resp70	int	Respuesta 70
Resp71	int	Respuesta 71
Resp72	int	Respuesta 72
Resp73	int	Respuesta 73
Resp74	int	Respuesta 74
Resp75	int	Respuesta 75
Resp76	int	Respuesta 76
Resp77	int	Respuesta 77
Resp78	int	Respuesta 78
Resp79	int	Respuesta 79
Resp80	int	Respuesta 80
RespFin	Nvarchar	Tipo de Aprendizaje obtenido
OrientMotR1	int	Respuesta 1 Cuestionario 2
OrientMotR2	int	Respuesta 2 Cuestionario 2
OrientMotR3	int	Respuesta 3 Cuestionario 2
OrientMotR4	int	Respuesta 4 Cuestionario 2
OrientMotR5	int	Respuesta 5 Cuestionario 2
OrientMotR6	int	Respuesta 6 Cuestionario 2
OrientMotR7	int	Respuesta 7 Cuestionario 2
OrientMotR8	int	Respuesta 8 Cuestionario 2
Resultado	Nvarchar	Respuesta del cuestionario

Las celdas marcadas son las afectadas en el segundo cuestionario.

A3.4 Programación del Escenario de Estructura de Control

Introducción del Escenario

Este escenario está programado con Flash y se utiliza actionscript para la programación. En este escenario se diseñaron 10 fotogramas para el caso de estudio estructuras de control

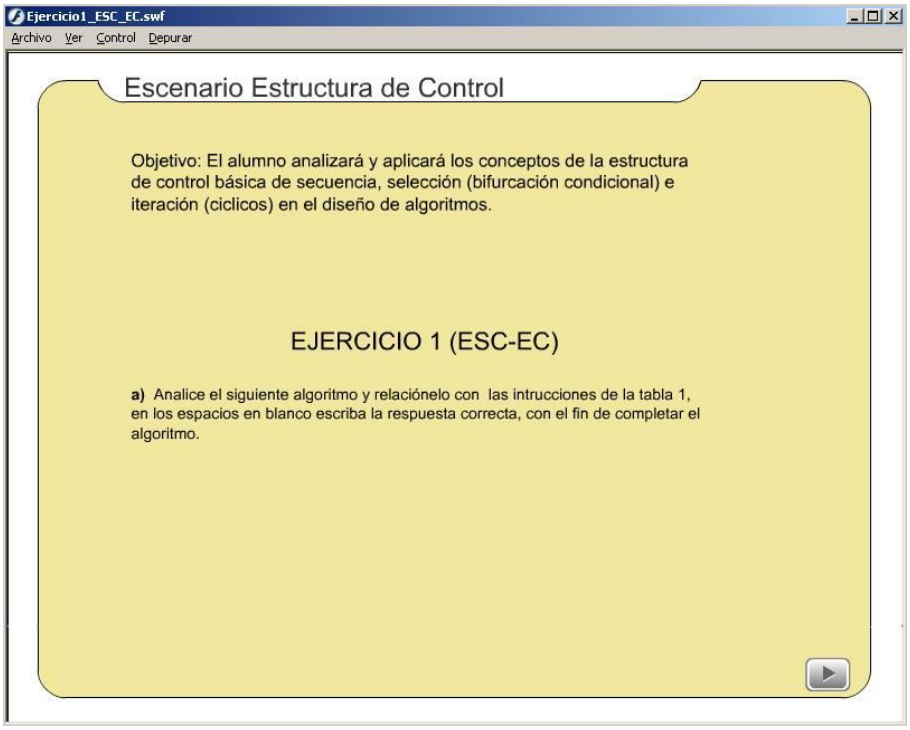


Figura A3.9 Pantalla de Presentación del Escenario de Estructuras de Control.

Código de Programación del la Página de Presentación del Escenario

En esta parte se inicializan las variables que se van a utilizar durante el ejercicio, además de las funciones y cargamos los datos del Archivo DatosEC.xml, el cual está contenido en una Carpeta de la raíz llama “Archivos”.

```
stop();

function cargarXml()
```

```
{
        objXml.load("Archivos/DatosEC.xml");
}
x_min=30;
min=0;
//cronometro
momentoInicio=0;
n_aciertos=0;
n_errores=0;
n_intentos=0;
n_fallas=0;
//vector de inferencia
Interes=1;
Deseo=1;
Ayuda=0;
Estrategia=0;
Interrupcion=0;
Renuncia=0;
Aprendizaje=0;
TiempoInact=1;
Errores=0;
//Errores
Graves=0;
Fatales=0;
Leves=0;
//objetivo de aprendizaje
var Nombre,Objetivo,Chaea;
//función de envio
function Vector()
```

```
{
	LblInteres.text=_root.Interes;
	LblDeseo.text=_root.Deseo;
	LblAyuda.text=_root.Ayuda;
	LblEstrategia.text=_root.Estrategia;
	LblInterrupcion.text=_root.Interrupcion;
	LblRenuncia.text=_root.Renuncia;
	LblAprendizaje.text=_root.Aprendizaje;
	LblTiempo.text=_root.TiempoInact;
	LblError.text=_root.Errores;
}
//Inicializamos nuestro xml
objXml=new XML();
/*Esta línea es EXTREMADAMENTE necesaria
  Es la que nos permite colocar espacios entre etiquetas
  Sin ella, tendríamos que mantener completamente pegado nuestro XML
  Y se pasearía mal dentro de Flash
*/
objXml.ignoreWhite=true;
//
objXml.onLoad=function(exito)
{
	if(exito)
	{
		_root.Nombre=objXml.firstChild.childNodes[0].firstChild.nodeValue;
		_root.Objetivo=objXml.firstChild.childNodes[1].firstChild.nodeValue;
		_root.Chaea=objXml.firstChild.childNodes[2].firstChild.nodeValue;
	}
	else
```

```
            _root.Nombre="Error al cargar XML";

        }
        cargarXml();
```

Código del Archivo Datosec.Xml

```
<?xml version="1.0" encoding="utf-8" ?>
- <Datos>
  <Nombre>Victor Gomez Quintero</Nombre>
  <Objetivo>Externos</Objetivo>
  <Chaea>Reflexivo</Chaea>
    </Datos>
```

En la página Escenario.aspx, realizamos la escritura del archivo DatosEc.xml.

Código de Programación de la Página Escenario.aspx

```
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class Escenario : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
```

```
{
    if (!IsPostBack)
    {

        if (Request.Cookies["userID"] != null)
        {
            LblID.Text = Server.HtmlEncode(Request.Cookies["userID"].Value);
        }
        else
            Server.Transfer("Default.aspx");
    }
    Cuestionarioip nom = new Cuestionarioip();
    nom.ID = Convert.ToInt32(LblID.Text);
    string Nombre = nom.ObtenerNombre();
    //obtenemos el objetivo de aprendizaje
    Cuestionarioip objetivo = new Cuestionarioip();
    objetivo.ID = Convert.ToInt32(LblID.Text);
    string Objetivo = objetivo.ObjetivoAprendizaje();
    //Obtenemos resultado de Chaea
    Cuestionarioip TipoApren = new Cuestionarioip();
    string Chaea = TipoApren.TipoAprendizaje();
    //generamos el xml
    Registroxml datos = new Registroxml(Server.MapPath("Archivos/DatosEC.xml"));
    datos.Nombre = Nombre;
    datos.Objetivo = Objetivo;
    datos.Chaea = Chaea;
    datos.CargarXML();
}
protected void CmdSiguiente_Click(object sender, EventArgs e)
```

```
    {

    }
}
```

Utilizamos el objeto Registroxml para realizar la escritura del archivo.

Código de Programación del Registroxml.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Xml;
using System.Text;
/// <summary>
/// Descripción breve de Registroxml
/// </summary>
public class Registroxml
{
    string recibe;

    public Registroxml(string strArchivo)
    {
```

```
    recibe = strArchivo;


 }
private string _Nombre;
public string Nombre
{
    get { return _Nombre; }
    set { _Nombre = value; }
}

private string _Objetivo;

public string Objetivo
{
    get { return _Objetivo; }
    set { _Objetivo = value; }
}
private string _Chaea;

public string Chaea
{
    get { return _Chaea; }
    set { _Chaea = value; }
}
public void CargarXML()
{
    XmlTextWriter escritor = new XmlTextWriter(recibe,Encoding.UTF8);
    escritor.Formatting = Formatting.Indented;
    escritor.WriteStartDocument();
```

```
        escritor.WriteStartElement("Datos");
        //inicia elemento Nombre
        escritor.WriteElementString("Nombre", _Nombre);
        escritor.WriteElementString("Objetivo", _Objetivo);
        escritor.WriteElementString("Chaea", _Chaea);
        escritor.WriteEndElement();
        escritor.WriteEndDocument();
        escritor.Close();
    }
}
```

A continuación de la pantalla de presentación del Escenario, se muestra el ejercicio, tal como se ve en la Figura 9.

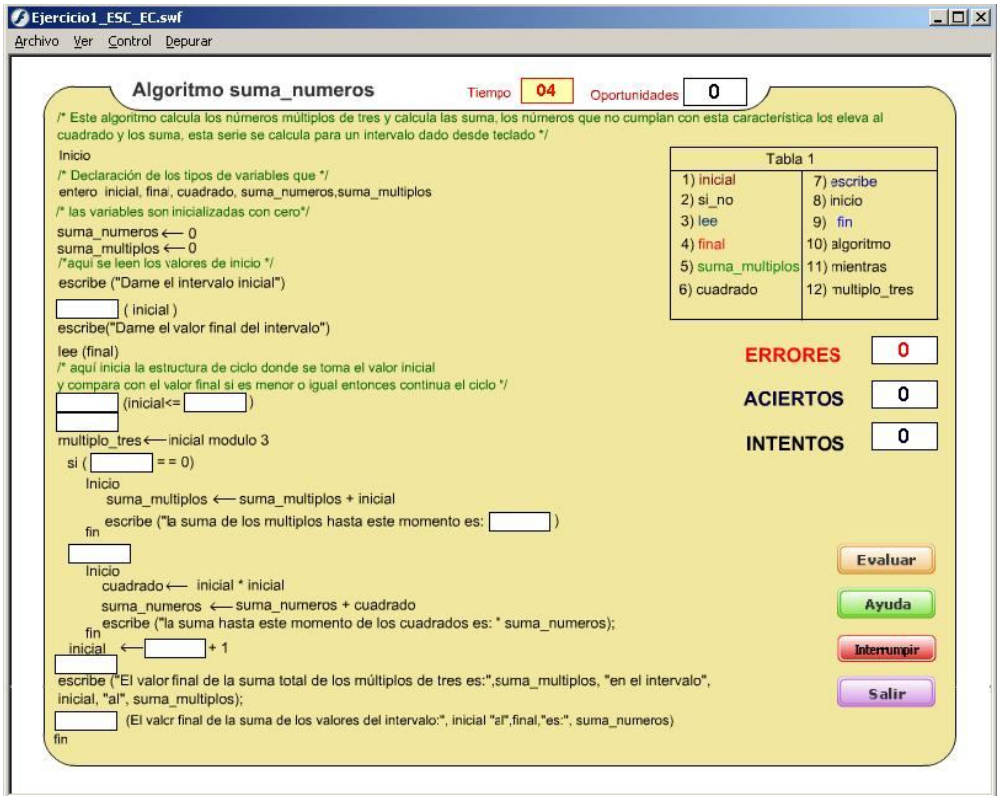


Figura A3.10 Pantalla del Escenario de Estructuras de Control.

El ejercicio cuenta con dos formas de control por medio de los botones y por Reloj. Cada control es independiente del otro y no tienen conexión alguna.

Control por Reloj

Cuando se entra a esta sección comenzara a correr un reloj Tiempo, que nos proporciona 30 segundos para responder cualquier de las pregunta, sino se responde ninguna de las pregunta en este tiempo incrementará una variable llamada Oportunidades, a los primeros 30 segundos nos envía un mensaje invitando al alumno a responder el ejercicio.

Se tiene 3 oportunidades para responder alguna pregunta, esto es 90 segundos para responder una pregunta, en caso de no hacerlo nos enviará al fotograma 7, donde dos despliega que hemos cometido un error grave y nos remitirá al inicio (fotograma 1) del ejercicio.

Utilizamos el Evento de Clip onClipEvent(enterFrame), este evento nos permiten que las acciones sean activadas en el momento de que una instancia de Clip entra o sale de la escena, utilizamos la función getTimer() que nos devuelve (en milisegundos) el tiempo transcurrido desde que la actividad (película) se abre.

Código de Programación del Reloj

```
onClipEvent(enterFrame)
{
        x_seg=(int(getTimer()/1000))-_root.momentoInicio;//segundos en escena
        seg=(x_seg-_root.x_min)+30;

        //En esta sección vamos a revisar los intentos que llevamos
        //tiempo de ocio
        if(_root.n_intentos==0)
                {
                        _root.LblAviso2.text=" ";
                }
```

```
		if(_root.n_intentos==1)
		{


		}
		if(_root.n_intentos==2)
		{
			_root.LblAviso2.text="¡Hey! Es tiempo de comenzar a trabajar";
			_root.TiempoInact=1;
		}
		if(_root.n_intentos==3)
		{
			_root.gotoAndPlay(7);
		}


if(x_seg==_root.x_min)
{//if que revisa cuando segundos es igual a 30
	seg=0;
	_root.x_min=_root.x_min+30;
	++_root.n_intentos;
}//termina if de 30
else if(x_seg>_root.x_min)
	{ //Revisamos si es mayor a 30
	 seg=0;
	 _root.x_min=x_seg+30;


	}
	_root.LblOport.text=_root.n_intentos;


if((seg)<10)
```

```
	{//if en caso de segundos sean menores a 10 despliege un cero a la izq.
		seg="0"+ seg;
	}//termina if de segundos
	_root.MomentoInicio=0;
	_root.Reloj.Segundos.text=seg;
}
```

También se utiliza un el siguiente código en el fotograma de la escena donde se encuentran nuestras cajas de textos para verificar que cuando cambia el valor de cualquiera de las cajas de textos cambiamos el valor de Oportunidades a cero y despliegue de las cajas de textos.

```
LblErrores.text=_root.n_errores;
LblAciertos.text=_root.n_aciertos;
LblOport.text=_root.n_intentos;
LblFallas.text=_root.n_fallas;
LblNombre.text=_root.Nombre;
//Realizamo revision de las cajas de textos
Resp01.onChanged=function()
{
	_root.n_intentos=0;
}
Resp02.onChanged=function()
{
	_root.n_intentos=0;
}
Resp03.onChanged=function()
{
	_root.n_intentos=0;
}
```

```
Resp04.onChanged=function()
{
	_root.n_intentos=0;
}
Resp05.onChanged=function()
{
	_root.n_intentos=0;
}
Resp06.onChanged=function()
{
	_root.n_intentos=0;
}
Resp07.onChanged=function()
{
	_root.n_intentos=0;
}
Resp08.onChanged=function()
{
	_root.n_intentos=0;
}
Resp09.onChanged=function()
{
	_root.n_intentos=0;
}
Resp10.onChanged=function()
{
	_root.n_intentos=0;
}
Resp11.onChanged=function()
```

```
{
	_root.n_intentos=0;
}
Resp12.onChanged=function()
{
	_root.n_intentos=0;
}
Resp13.onChanged=function()
{
	_root.n_intentos=0;
}
Resp14.onChanged=function()
{
	_root.n_intentos=0;
}
Resp15.onChanged=function()
{
	_root.n_intentos=0;
}
Resp16.onChanged=function()
{
	_root.n_intentos=0;}
```

Control por Botón

Este flujo de control se da a través de un botón Evaluar, Ayuda, Interrumpir y Salir ubicadas en la escena.

Código de Programación del Botón Evaluar.

```
on(press)
{
	_root.n_aciertos=0;
	_root.n_errores=0;
	if(_root.Resp01.length==0 || _root.Resp02.length==0 || _root.Resp03.length==0 ||_root.Resp04.length==0 || _root.Resp05.length==0 || _root.Resp06.length==0 || _root.Resp07.length==0 ||_root.Resp08.length==0 || _root.Resp09.length==0 || _root.Resp10.length==0)
	{
		_root.LblAviso.text="No ha contestado todas las preguntas";
		_root.n_fallas=0;
		_root.LblFallas.text=_root.n_fallas;

	}
	else
	{//inicia else principal
		if(Number(_root.Resp01.text)==3 || _root.Resp01.text=="lee")
		{
			++_root.n_aciertos;
		}
		else
		{
			++_root.n_errores;
			++_root.Leves;
		}
		if (Number(_root.Resp02.text)==11 || _root.Resp02.text=="mientras")
		{
			++_root.n_aciertos;
		}
```

```
else
{
	++_root.n_errores;
	++_root.Fatales;
}

if(Number(_root.Resp03.text)==4 || _root.Resp03.text=="final")
{
	++_root.n_aciertos;
}
else
{
	++_root.Graves;
	++_root.n_errores;
}
if(Number(_root.Resp04.text)==8 || _root.Resp04.text=="inicio")
{
	++_root.n_aciertos;
}
else
{
	++_root.Graves;
	++_root.n_errores;
}

if(Number(_root.Resp05.text)==12 || _root.Resp05.text=="multiplo_tres")
{
	++_root.n_aciertos;
}
```

```
else
{
	++_root.n_errores;
	++_root.Leves;
}
if(Number(_root.Resp06.text)==5 || _root.Resp06.text=="suma_multiplos")
{
	++_root.n_aciertos;
}
else
{
	++_root.n_errores;
	++_root.Leves;
}

if(Number(_root.Resp07.text)==2 || _root.Resp07.text=="si_no")
{
	++_root.n_aciertos;
}
else
{
	++_root.n_errores;
	_root.Fatales;
}
if(Number(_root.Resp08.text)==1 || _root.Resp08.text=="inicial")
{
	++_root.n_aciertos;
}
else
```

```
{
	++_root.n_errores;
	++_root.Leves;
}

if(Number(_root.Resp09.text)==9 || _root.Resp09.text=="fin" )
{
	++_root.n_aciertos;

}
else
{
	++_root.Graves;
	++_root.n_errores;
}
if(Number(_root.Resp10.text)==7 || _root.Resp10.text=="escribe")
{
	++_root.n_aciertos;
}
else
{
	++_root.n_errores;
	++_root.Leves;
}

//comienza las comparaciones para  ver el número de fallas
	if(_root.Fatales!=0)
	{
		_root.Estrategia=1;
```

```
            _root.gotoAndPlay(3);
}
else if(_root.Graves!==0)
 {
             _root.Estrategia=1;
             _root.gotoAndPlay(7);
            }
else if(_root.Leves!=0)
{
            _root.Estrategia=1;
            _root.LblAviso.text="¡Yo sé que puedes hacerlo!";
}

if(_root.n_fallas==2)
{
            _root.gotoAndPlay(6);
}
else
{
            if(_root.n_aciertos==10 && _root.n_fallas==0 )
            {
                        _root.Aprendizaje=1;
                        _root.Errores=0;
                        _root.gotoAndPlay(4);
            }
            if(_root.n_aciertos==10 && _root.n_fallas==1)
            {
                        _root.Aprendizaje=1;
                        _root.Errores=1;
```

```
                    _root.gotoAndPlay(5);
                }

                if(_root.n_aciertos!=10)
                {
                    _root.Errores=1;
                    ++_root.n_fallas;
                }

            }
        }

        _root.LblAciertos.text=_root.n_aciertos;
        _root.LblErrores.text=_root.n_errores;
        _root.LblFallas.text=_root.n_fallas;

}
```

En este código realizamos la verificación de que se contesten todas las preguntas, y evaluar si existe algún tipo de error fatal, grave o Leve.

- **Si tenemos un error fatal nos envía al fotograma 3.**
- **Si tenemos un error grave nos envía al fotograma 7.**
- **Si tenemos algún error leve nos permite seguir con el ejercicio. (Nos da una segunda oportunidad para su resolución).**

También verificamos si hemos contestado correctamente el ejercicio, cuantos aciertos y cuantos errores tenemos.

- **Si contestamos correctamente en la primera oportunidad nos manda al fotograma 4.**
- **Si contestamos correctamente en nuestra segunda oportunidad nos manda al fotograma 5.**

- **Si no contestas correctamente en la segunda oportunidad nos envía al fotograma 6.**

Código de Programación del Botón de Ayuda

```
on(press)
{
	_root.Estrategia=1;
	_root.Ayuda=1;
	_root.gotoAndPlay(8);
}
```

Las dos acciones que se ejercen son: La variable Ayuda y Estrategia del vector le asignamos el valor de 1. Y saltamos al fotograma 8.

Código de Programación del Botón Interrupción

```
on(release)
{
	_root.Interrupcion=1;
	_root.momentoInicio=int(getTimer()/1000);
	_root.gotoAndPlay(9);
}
```

Aquí se ejercen tres acciones:

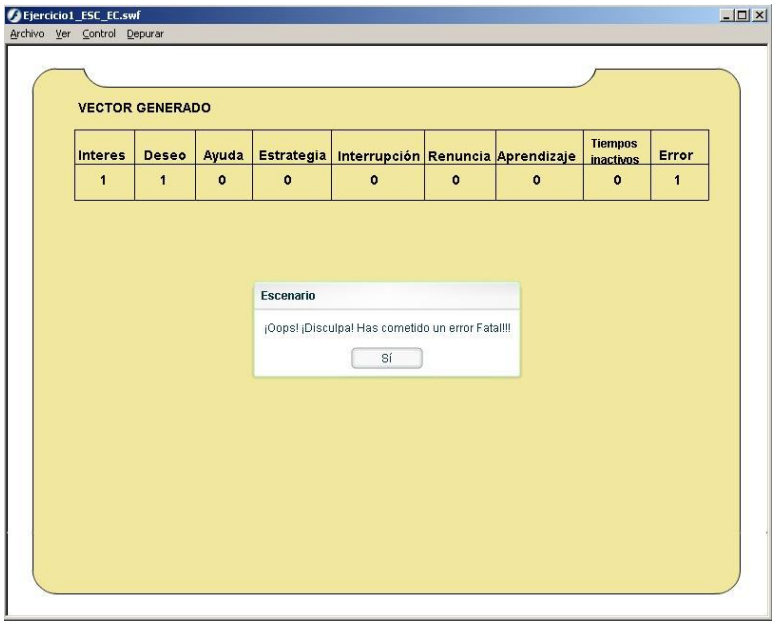
- Se pone la variable Interrupcion en 1.
- Tomamos el tiempo de la interrupción para utilizarla en el fotograma 9.
- Y saltamos al fotograma 9.

Código de Programación del Botón Salir

```
on(press)

on(press)

{

        _root.Renuncia=1;

        _root.gotoAndPlay(10);

}
```

Ejercemos dos acciones: Ponemos en 1 la variable Renuncia y saltamos al fotograma 10.

Fotograma 3



En este fotograma se llega cuando se ha cometido en error fatal al responder al responder el cuestionario.

Código de Programación del Fotograma 3

Mostramos los valores que tiene el vector y un cuadro de alerta y se cierra el escenario.

```
stop();

import mx.controls.Alert;

Vector();

alClicar = new Object();

        alClicar = function (evento)

        {       if (evento.detail == Alert.YES)

                        {

                                unloadMovie(_root);

                        }


        }

        Alert.yesLabel = "Sí";

        Alert.noLabel = "No";

        Alert.buttonWidth = 75;

        Alert.show("¡Oops! ¡Disculpa! Has cometido un error Fatal!!!", "Escenario",Alert.YES, this, alClicar, "Salir", Alert.OK);
```

Fotograma 4

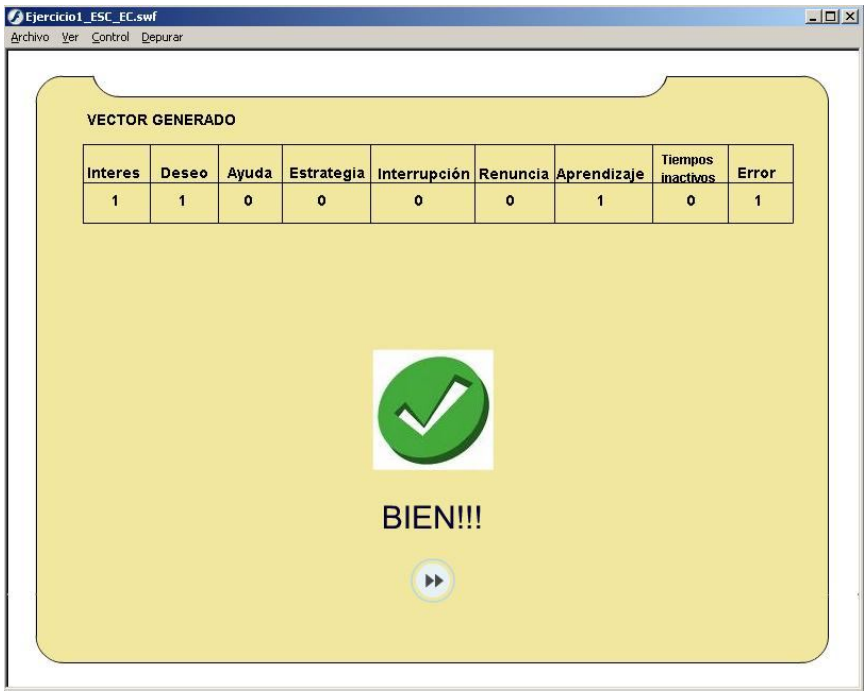
Cuando se contesta correctamente en la primera oportunidad nos envía a este fotograma.

Código de Programación del Fotograma 4

```
stop();

Vector();
```

Solo mostramos los valores del vector obtenido con la función Vector ().

Fotograma 5



Cuando llenamos correctamente el cuestionario en nuestra segunda oportunidad nos manda a este fotograma, al igual que el fotograma 4 solo mostramos el vector obtenido

Código de Programación del Fotograma 5

```
stop();

Vector();
```

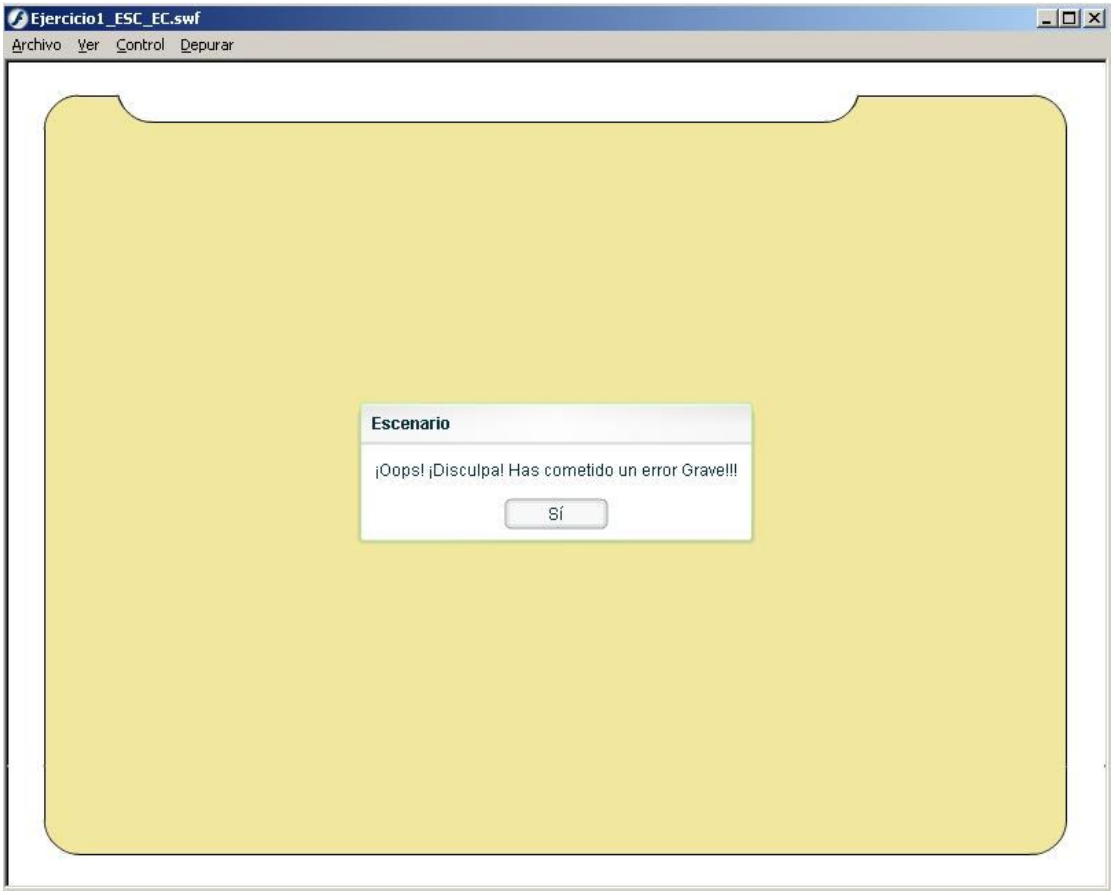
Fotograma 6

Cuando se comete el segundo error al llenar el cuestionario nos envía a este fotograma y mostramos el vector obtenido

Código de Programación del Fotograma 6

```
stop();

Vector();
```

Fotograma 7

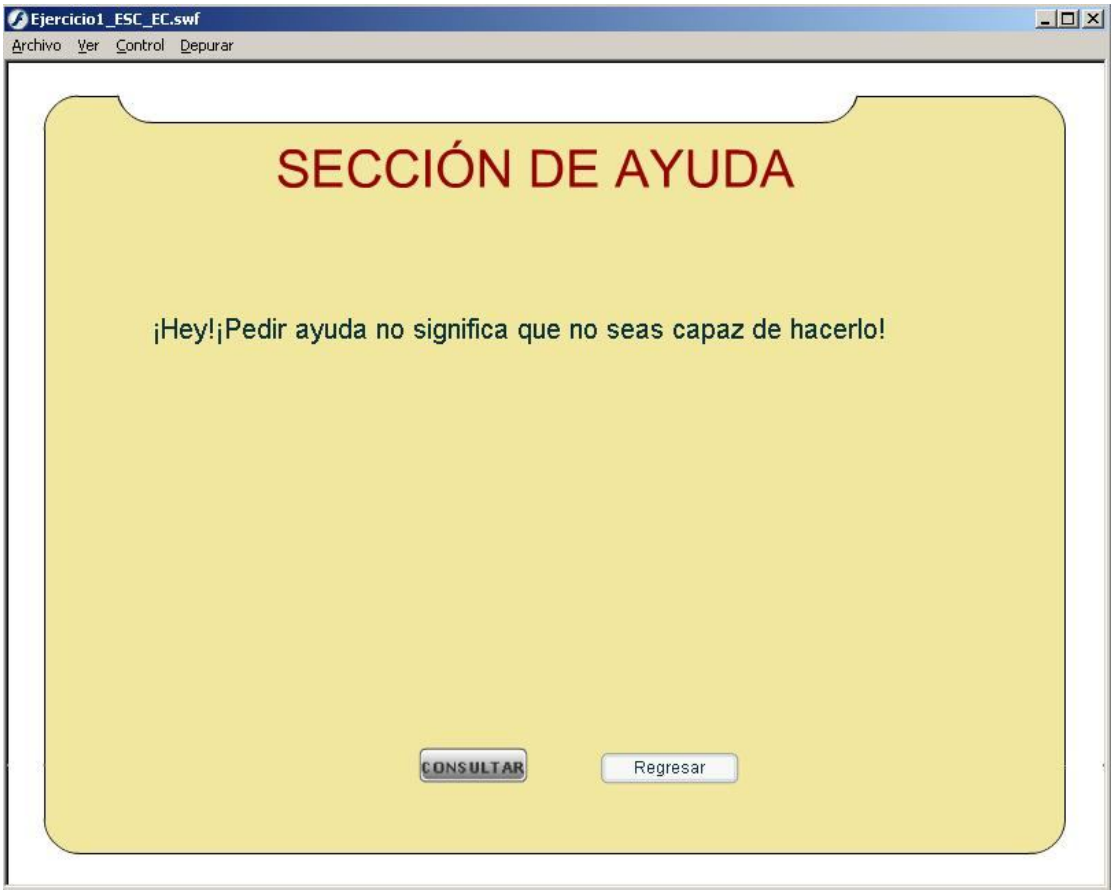


Código de Programación del Fotograma 7

```
stop();
import mx.controls.Alert;
alClicar = new Object();
        alClicar = function (evento)
        {       if (evento.detail == Alert.YES)
                        {
                                _root.gotoAndPlay(1);
                        }
```

```
}

Alert.yesLabel = "Sí";

Alert.noLabel = "No";

Alert.buttonWidth = 75;

Alert.show("¡Oops! ¡Disculpa! Has cometido un error Grave!!!", "Escenario",Alert.YES, this, alClicar, "Inicio", Alert.OK);
```

Fotograma 8



Código de Programación del Fotograma 8

```
function EstiloAyuda()
{
	_root.LblAyuda.setStyle("fontWeight",bold);
	_root.LblAyuda.setStyle("fontSize",20);
}
//revisamos que tipo de orientacion tenemos
if(_root.Objetivo=="Internos")
	{
		EstiloAyuda();
		_root.LblAyuda.text="!Buen momento para expresar dudas! Este tema no es del todo sencillo";
	}
else{
	if(_root.Objetivo=="Externos")
	{
		EstiloAyuda();
		_root.LblAyuda.text="¡Hey!¡Pedir ayuda no significa que no seas capaz de hacerlo!";

	}
	else
	{
		EstiloAyuda();
		_root.LblAyuda.text="No identifico tu tipo de orientación";
	}
 }
stop();
```

Código botón “Consultar”

```
on(press)
{
        getURL("http://aniei.org.mx/paginas/uam/CursoIP/curso_ip_09.html");
}
```

Código botón “Regresar”

```
on(click)
{
        _root.gotoAndStop(2);
        _root.n_intentos=0;
}
```

Fotograma 9

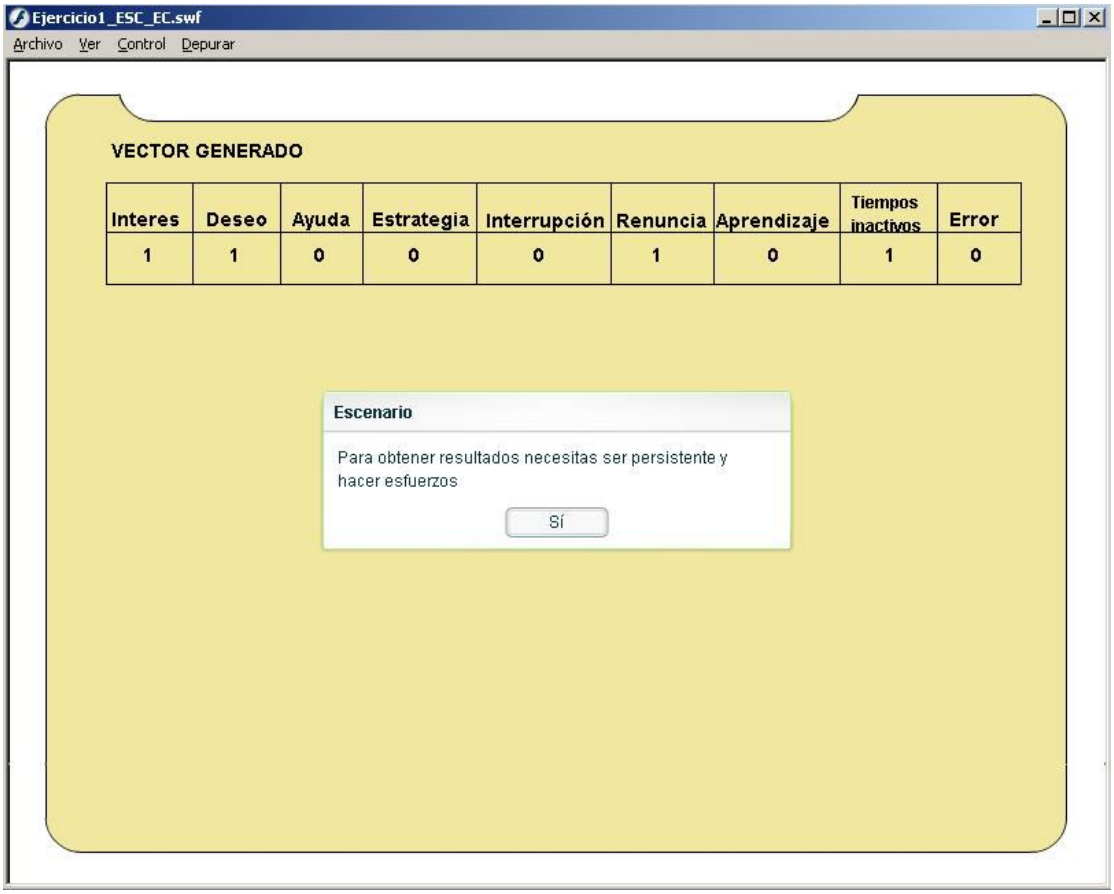
Código de Programación del Fotograma 9

```
stop();
function EstiloInt()
{
	_root.LblInterrupcion.setStyle("fontWeight",bold);
	_root.LblInterrupcion.setStyle("fontSize",20);
}
//comprobamos el estilo de aprendizaje
if(_root.Objetivo=="Internos")
	{
		EstiloInt();
		_root.LblInterrupcion.text="¡Que gusto ayudarte!";
	}
	else
	{
		if(_root.Objetivo=="Externos")
		{
			EstiloInt();
			_root.LblInterrupcion.text="¿Qieres más consejos?";
		}
		else
		{
			EstiloInt();
			_root.LblInterrupcion.text="NO detecto tu Objetivo de Aprendizaje";
		}
	}
```

Código Botón

```
on(press)
{
        _root.gotoAndStop(2);
}
```

Fotograma 10



Código de Programación del Fotograma 10

```
stop();

import mx.controls.Alert;


Vector();


alClicar = new Object();
        alClicar = function (evento)
        {       if (evento.detail == Alert.YES)
                        {
                                unloadMovie(_root);
                        }


        }
        Alert.yesLabel = "Sí";
        Alert.noLabel = "No";
        Alert.buttonWidth = 75;
        Alert.show("Para obtener resultados necesitas ser persistente y hacer esfuerzos", "Escenario",Alert.YES, this, alClicar, "Salir", Alert.OK);
```