INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS ESTADO DE MÉXICO

DIVISIÓN DE GRADUADOS E INVESTIGACIÓN
PROGRAMA DE GRADUADOS EN INGENIERÍA

El paradigma de objetos: Un enfoque de la alta tecnología en la educación

TESIS PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO ACADÉMICO DE **MAESTRO EN CIENCIAS CON ESPECIALIDAD EN SISTEMAS DE INFORMACIÓN**

presentó: **JORGE LUIS FERNANDO GÓMEZ FRAGOSO**

BIBLIOTECA

Siendo integrado el jurado por:

Humberto Cárdenas Anaya

Ralf Eder Lange

Joaquín Méndez Almaguer

Rodrigo Polanco Bueno

ÍNDICE

Resumen i

Introducción ii

1. Objetivos 1

2. Análisis del problema 2

2.1. Antecedentes del problema 2

2.2. Exposición del problema 4

3. Metodología de elaboración 8

4. Programación orientada a objetos en los planes de estudio de las carreras de sistemas computacionales 10

4.1. Introducción 10

4.2. Referencias 12

4.2.1. Universidades 12

4.2.2. Libros de texto 14

4.2.3. Encuestas 16

4.2.4. Otros estudios 18

4.2.5. Psicología del aprendizaje 21

4.3. Selección y definición de los conceptos fundamentales ... 24

4.4. Metodología de enseñanza 33

5. Un tutorial para la enseñanza de la POO 44

5.1. Aplicación de la alta tecnología en la educación 44

5.2. Características de un buen tutorial 49

5.3. Selección de hardware y software 53

5.4. Diseño e implementación 55

6. Discusión 59

7. Conclusiones 61

Bibliografía 65

Apéndices 69

Apéndice A. Programas universitarios 70

Apéndice A-I. Tabla comparativa 71

Apéndice A-II. Programa de James McKim 73

Apéndice A-III. Programa de la Universidad Estatal de Oklahoma 75

Apéndice B. Tabla comparativa de libros de texto 77

Apéndice C. Encuesta 81

Apéndice C-I. Formato de la encuesta 82

Apéndice C-II. Tabla de resultados de la encuesta 85

Apéndice D. Diagrama para la selección de medios que ayuden al aprendizaje 90

Apéndice E. Recursos 92

Apéndice E-I. Humanos 93

Apéndice E-II. Técnicos 96

Resumen

La programación orientada a objetos (POO) y la influencia de la alta tecnología en la educación son temas de mucha actualidad y creciente interés por parte de los investigadores de las ciencias computacionales y la pedagogía.

En este trabajo se estudia la situación actual de la enseñanza de la programación orientada a objetos, que incluye varios programas de estudio de la materia, libros de texto sobre el tema, trabajos realizados sobre la forma de impartirla y la preferencia de algunos alumnos por una metodología de enseñanza especifica. Con base en lo anterior y en un análisis del proceso humano de aprendizaje se propone una metodología con la forma óptima de enseñar programación orientada a objetos.

Se analiza por otro lado, la utilidad de elementos de tecnología avanzada en el proceso enseñanza-aprendizaje y a partir de ellos las características que debe tener un buen tutorial. Al final se presenta una propuesta con la manera adecuada de fusionar, la forma de enseñar POO y la alta tecnología en la educación a través de un tutorial, teniendo como resultado el prototipo de lo que es una herramienta con multimedia para el aprendizaje de los conceptos fundamentales del paradigma de objetos.

Introducción

Una de las mayores preocupaciones que se tienen entre los docentes es la búsqueda de los programas de estudio ideales para las materias que se imparten, al mismo tiempo con la incorporación de las nuevas tecnologías para brindarle a los alumnos un conocimiento acorde con la realidad. Esta tarea es más difícil aún si la materia es de algún tema novedoso como ocurre casi siempre en el área de sistemas, ya que apenas se están consolidando sus aspectos más importantes y todavía se realizan estudios sobre él. Así se encuentra mucha gente de sistemas computacionales que elabora temarios para enseñar las teorías de su área de especialidad.

Por otro lado, una de las principales áreas de investigación para los psicólogos es la docente, existen muchos estudios para determinar conductas en los estudiantes que ayudan a desarrollar herramientas o procesos que facilitan el aprendizaje de algún tema. Desde hace aproximadamente cincuenta años se han desarrollado tutoriales como lo más parecido a una herramienta educativa y a través de los años su progreso ha sido notable, sin embargo, los resultados todavía no son muy buenos comparados con el que para muchos es el mejor tutorial que existe: el simulador de vuelo.

Aunque el objetivo entre las dos áreas mencionadas anteriormente, la de sistemas y la educativa, es el mismo: optimizar el aprendizaje de los temas, estos esfuerzos casi nunca están unidos, y como si fuera poco, ni los unos ni los otros aprovechan - o dominan - los avances tecnológicos que mejorarían aún más el proceso enseñanza-aprendizaje.

Retomando un caso particular, en la película "Parque Jurásico" de Steven Spielberg, el dueño del parque, cuando lo construyó, tenía como objetivo el hacer que todas las personas que lo visitaran fueran partícipes de una experiencia real, divertida e inolvidable. De manera

similar en un local de video juegos se nota que los jóvenes pasan horas jugando emocionados sin fatigarse, esto para muchos representa un reto y les proporciona placer. Así deberían ser todos los tutoriales, con un ingrediente adicional ... el aprendizaje de algún tema especial para la formación profesional.

Los programas de estudio y los tutoriales son las ideas que conducen esta tesis que trata sobre la forma adecuada de impartir la materia de programación orientada a objetos (POO) a los estudiantes de la carrera Licenciados en Sistemas Computacionales Administrativos (LSCA) del Instituto Tecnológico y de Estudios Superiores de Monterrey; Campus Estado de México (ITESM-CEM) y como apoyarse en la tecnología computacional avanzada para su aprendizaje, esto es, utilizar lo último en herramientas computacionales para optimizar el proceso enseñanza-aprendizaje.

Inicialmente se pensó en el desarrollo de una herramienta para la programación orientada a objetos, como base para la enseñanza del curso Tópicos I (Programación Orientada a Objetos). Sin embargo, en el transcurso del análisis de esta idea, se cayó en cuenta que para poder utilizar adecuadamente una herramienta de programación se deben dominar primero los conceptos fundamentales que se desean aplicar. Motivo por el cual ahora se propone una metodología a ser seguida en clase para la enseñanza de dichos conceptos y la forma que debe llevar un tutorial que la respalde. El reto es la conjunción de tres áreas muy interesantes: la docente, el paradigma de objetos y la tecnología.

Esta tesis se encuentra dividida en siete capítulos, el primero de ellos, los objetivos de la tesis, expone de una forma precisa qué se va a obtener con este trabajo, para así conocer desde el inicio su razón de ser.

El segundo capítulo, "Análisis del problema", tiene dos secciones, en la primera se describe qué estudios o esfuerzos se hacen similares a éste, y en la otra, se brinda una descripción del problema y se muestra la utilidad del mismo para comprobar si realmente soluciona un problema.

En el siguiente capítulo, "La metodología", se exponen los pasos que se siguieron para realizar este trabajo, las etapas y cómo se llevaron a cabo, los problemas encontrados y cómo fueron solucionados y todo aquello que amerita ser considerado dentro del proceso de elaboración de la tesis.

"La programación orientada a objetos", es el siguiente capítulo; se encuentra dividido en cuatro secciones, la primera presenta una introducción a los objetos; la segunda contiene una exposición de los resultados obtenidos en la investigación de otros trabajos hechos sobre este tema que incluye, una encuesta de opinión para alumnos, un estudio bibliográfico y de universidades de la Ciudad de México; además para tener un fundamento educativo se elaboró un análisis de la forma de aprendizaje humano. La tercera sección presenta los conceptos fundamentales con su(s) definición(es) ideal(es) y por último la sección que contiene la propuesta para la enseñanza de la programación orientada a objetos.

El capítulo quinto, "El tutorial", muestra cómo se ha visto influenciada la educación con la tecnología computacional y cuál es su utilidad, la forma en que debe ser desarrollada una herramienta ideal de aprendizaje, la selección del hardware y software para realizarla y las consideraciones de diseño e implementación del tutorial con base en la propuesta del capítulo anterior.

El siguiente capítulo, el sexto, es una discusión, en donde se plantean las ventajas y áreas de oportunidad del trabajo expuesto. Por último, en el capítulo restante, se exponen las conclusiones obtenidas con el trabajo, si se lograron los objetivos o no y por qué.

En los apéndices se encontraran la encuesta y las tablas con sus resultados, las tablas comparativas de los programas universitarios y libros de texto, algunas gráficas representativas y los recursos que se necesitaron para llevar a cabo el trabajo, este comprende dos secciones, la primera referente al número y características de las personas involucradas, y la segunda, los recursos técnicos, hardware y software básicamente.

1. Objetivos

La transmisión de los conceptos fundamentales de la programación orientada a objetos es el objetivo principal de este trabajo, ya sea en las aulas de clase o a través de un tutorial. Para esto se considera:

- Identificar y definir los conceptos fundamentales de la programación orientada a objetos.
- Determinar el orden en que deben ser enseñados dichos conceptos para un óptimo aprendizaje de los mismos.
- Seleccionar el(los) lenguaje(s) y tipos de ejemplos y proyectos adecuados para facilitar el proceso enseñanza-aprendizaje.
- Determinar y evaluar el papel de la alta tecnología en la educación.
- Identificar las características de un buen tutorial.
- Seleccionar la tecnología necesaria para optimizar el proceso enseñanza-aprendizaje.
- Hacer el modelo de un tutor de programación orientada a objetos.

Como resultado se espera obtener tanto una propuesta con la metodología adecuada para la enseñanza de la programación orientada a objetos, como el modelo de una herramienta parecida a lo que se conoce tradicionalmente como tutorial, que enseñe de la forma más adecuada según los resultados de esta investigación y apoyándose en la alta tecnología, los conceptos fundamentales de la programación orientada a objetos. Este producto podrá ser utilizado por el profesor en la materia de Tópicos I como guía para impartir su clase y soporte para facilitar el aprendizaje de los alumnos, y por los alumnos como un asesor permanente.

2. Análisis del problema

2.1. Antecedentes del problema

Como profesor en el Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Estado de México (ITESM-CEM) se tiene semestralmente la responsabilidad de mantener actualizado el temario del curso que se va a impartir. Para ayudar a los profesores en su trabajo, el ITESM-CEM imparte un Diplomado en Habilidades Docentes con una parte enfocada al diseño de objetivos instruccionales generales y específicos basados en taxonomías del dominio cognoscitivo, afectivo y psicomotor, mostrando así la importancia que tiene la elaboración de un currículum para cada materia.

Con esto se obtiene una buena definición de qué, cómo y hasta dónde se espera el desarrollo del alumno en cada materia o tema, pero no del contenido que se debe dar y el orden a seguir para lograr de la mejor manera el aprendizaje de la materia. Si se toma en cuenta la selección del contenido y su asociación al objetivo de la clase o curso, no se garantiza de todas formas que ese objetivo sea el adecuado para tal contenido.

En la materia de Programación Orientada a Objetos en particular, los señores James McKim ([MCK 92] y [MCK 93]) y Bertrand Meyer ([MEY 93]) expertos en el tema han hecho propuestas bastante interesantes y serias de cómo se debe impartir esta materia, sin embargo, no son estándares a nivel universitario y algunas de sus indicaciones no aplican a nuestro caso, como ocurre con la de McKim válida para estudiantes de maestría y la de Meyer que tiene entre sus sugerencias que se imparta con una duración de un año y como primera materia de programación. Aunque los

trabajos de estos autores no aplican en su totalidad, sí contienen muchos otros tópicos válidos y útiles que serán mencionados a su debido tiempo.

Los anteriores no son los únicos autores que han presentado trabajos a ese respecto, entre otros se encuentran D'Souza ([DSO 93]) y Wu ([WU 93]), con más análisis teóricos de las situaciones que se deben enfrentar en la enseñanza, más que una presentación detallada del curso, sin embargo poseen aportes muy valiosos como se verá más adelante.

Los psicólogos educadores por otro lado, desde 1954 con Skinner y sus estudios sobre instrucción programada, han buscado crear una herramienta ideal para la enseñanza. Esta primera aproximación fue fundamental para los primeros tutoriales en forma de libros de texto, algunas de sus ideas prevalecen y la evolución de esas teorías es la base de los actuales tutoriales. También se ha buscado la forma para elaborar la currícula de las materias que se van a impartir, siendo el método más común la selección de los contenidos por parte de un grupo de expertos en el tema.

Afortunadamente para los usuarios de tutoriales y de este trabajo, con la evolución tecnológica y el auge de las interfases gráficas se han ido entrelazando los estudios del comportamiento y razonamiento humano a la forma y medios que deben tener las aplicaciones para que los usuarios sean más productivos, en este caso, que aprendan mejor.

En investigaciones recientes Chen ([CHE 91]) y Kuboni ([KUB 93]) hacen propuestas acerca de cómo se deben elaborar las herramientas de ayuda educativa dando las consideraciones necesarias para obtener un buen resultado, el primero basado en modelos del profesor, alumno y aprendiz muestra cómo influyen las

computadoras en la sociedad; el segundo, más enfocado a la secuenciación de los contenidos, sugiere manejar los temas de lo concreto a lo abstracto e ir colocando puntos de interés a través de todo el programa.

En junio de 1993 tuvo lugar en Orlando, Florida (E.U.) la ED-MEDIA 93, la primera conferencia mundial en multimedia e hipermedia educacional donde se presentaron trabajos y estudios en el área, que sirven para notar su situación en el mundo y mostrar la importancia adquirida por estas dos ramas en la educación. Se pudo apreciar cómo la multimedia es una buena alternativa de soporte a la educación, entre los resultados del evento se destacan trabajos para la elaboración de salas con equipo computacional en red y multimedia como recurso audiovisual, para hacer a los alumnos más partícipes de la clase.

2.2. Exposición del problema

El problema fundamental es determinar qué temas se deben impartir y en qué forma en la materia de programación orientada a objetos a los licenciados en sistemas de computación administrativa.

Las características que tiene esta materia son:

- Se imparte a alumnos de sexto semestre de formación profesional
- Es la quinta materia de programación que reciben
- Traen formación con programación estructurada y ciclo de vida tradicional
- No han cursado lógica matemática ni matemáticas computacionales
- Son grupos de treinta alumnos aproximadamente

Comprobar si estas características son buenas o malas no cae dentro de los alcances de esta tesis, pero sí sirven para centrar la investigación en los detalles que las satisfagan.

Básicamente se debe tener en cuenta algunos puntos esenciales como son el hecho de que son alumnos con experiencia en programación y poco énfasis en investigación, llegar con un nuevo estilo de programación implica romper los esquemas que tienen trazados lo cual demandaría enseñarles desde el principio todo. No se puede ser muy detallista o personalizar la enseñanza porque los grupos son muy heterogéneos y grandes, y tampoco se pretende enseñar toda la teoría matemática que está detrás del paradigma de objetos porque no tienen las bases para entenderla.

Frente a este panorama, parte de la dificultad radica en que no existe un temario estándar para impartir la materia, ya que para un mismo concepto existen varias definiciones que en algunos momentos representan cosas distintas, no hay un modelo del orden y metodología con el que se deben transmitir los temas, existen varios lenguajes orientados a objetos (LOO) con los cuales trabajar y no se cuenta con un conjunto de ejemplos o proyectos que faciliten la comprensión de la programación orientada a objetos. Adicionalmente, de las publicaciones que tratan esta problemática ninguna aplica a las características que aquí se tienen.

La selección de los temas que se van a impartir, es una tarea aparentemente sencilla ya que en casi todas las publicaciones y programas se tocan los mismos temas, salvo en algunos libros como los referentes al lenguaje Pascal orientado a objetos que no incluyen el concepto de clase ya que esto no aplica en este lenguaje como tal, así que bastaría escoger los temas más comunes. Pero, una vez seleccionados los conceptos que se deben dar, surge una complicación, cuál es el significado que se debe utilizar para cada concepto ya que según el autor y el tipo del libro de texto se tiene uno diferente.

Ahí no termina el problema, al impartir una materia de programación faltaría un elemento importantísimo, el lenguaje, hay una gran familia de estos en el mercado, divididos en híbridos, puros, orientados a objetos y basados en objetos, pero la selección del o los lenguajes adecuados también es parte de este trabajo. Otro problema que se deriva de los anteriores es el tipo de ejemplos y proyectos que se van a utilizar para facilitar el aprendizaje de acuerdo al perfil de los alumnos que se tienen.

Se debe considerar también la secuenciación de los conceptos fundamentales, ya que el orden puede afectar la comprensión de los temas, escoger una metodología - ya sea acorde a los objetos o una general como por ejemplo de lo concreto a lo abstracto o de lo familiar a lo no familiar - o proponerla. Para esto es fundamental la opinión de los psicólogos, sobre todo para que se conozca la forma de aprender de las personas y así adaptar la metodología.

El último factor que se debe afrontar es el del medio de transmisión del conocimiento. Para aplicar la metodología tenemos como medios una clase magistral con un profesor de tiempo completo, con maestría en su área de especialidad, lo cual se debe a una condición del Instituto, y el desarrollo de un tutorial, aprovechando que la tecnología nos ofrece la oportunidad de hacer cada vez más cosas, que ayuda en este proceso de enseñanza-aprendizaje. Aquí se cuestiona la forma de optimizar el proceso con la ayuda de alta tecnología. Con este planteamiento hay también que determinar en qué plataforma se podría desarrollar y con qué aplicaciones para brindarle un mejor servicio a los alumnos.

En particular, esta tesis provee a los profesores de la materia de programación orientada a objetos en el ITESM-CEM la forma para impartir sus clases de la manera más adecuada, y en general a los profesores de la materia de otras instituciones ofrece

una guía para su curso en donde con unas pocas modificaciones podrán obtener la que se ajuste a sus necesidades; también brinda las bases para justificar la utilización de la multimedia como una herramienta de apoyo a la docencia y el modelo del ideal de un tutorial para la enseñanza de la programación orientada a objetos.

3. Metodología

Para cumplir con los objetivos del trabajo este se ha dividido en dos partes principales. Una que se encargó de la parte del paradigma de objetos, esto es, selección de los conceptos adecuados, sus definiciones, secuenciación, metodología de enseñanza, determinación del lenguaje, ejemplos y proyectos.

Para abordar esta parte se comenzó por la búsqueda de trabajos similares de otras personas en revistas especializadas en programación orientada a objetos y educación para conocer el por qué de sus planteamientos y poderlos comparar y adaptar. Después se hizo una comparación entre algunos libros sobre el tema para encontrar lo que tenían en común y detectar una secuencia lógica en sus temas de tal forma que se pudieran seguir. También se hicieron averiguaciones en algunas universidades de la Ciudad de México para saber cuáles eran sus programas de estudio y luego se compararon entre sí para establecer un estándar, se incluyeron adicionalmente dos programas de estudio. Por último y para no desvincular a los clientes, los alumnos de sistemas computacionales, se aplicó una encuesta para conocer su punto de vista y satisfacerlos lo mejor posible.

La segunda parte del trabajo, consiste en la elaboración del modelo del tutorial, para lo cual se analizan estudios sobre la utilidad e influencia de la alta tecnología en la educación, sus interacciones y se consiguió información sobre la forma de aprender y percibir de los seres humanos para identificar la forma que debe tener un tutorial ideal. Para cerrar un poco el espacio de estudio se tomó como referencia el área de la multimedia.

Después se hizo la selección del hardware y software sobre el que se debe hacer el tutorial tomando como base lo obtenido en la etapa descrita anteriormente y como referencia

lo que había en el Instituto, para esto se averiguó con qué se contaba en el ITESM-CEM y se evaluaron las posibles alternativas.

Lo último en realizarse fue el diseño y la elaboración del prototipo del tutorial, que incluye la creación de cada una de sus partes. Para esto se trabajó básicamente el aprendizaje de las herramientas, el diseño de cada una de las pantallas, que de acuerdo a lo que se investigó es lo ideal, la elaboración del guión o secuenciación de las pantallas, la implantación de los guiones y videos necesarios y la incorporación del sonido.

4. Programación orientada a objetos

4.1. Introducción

El paradigma de objetos es un conjunto de teorías y reglas que sirven para representar el mundo o problema lo más parecido posible a la realidad, con base en elementos naturales como objetos y clases, facilitando la comunicación entre los desarrolladores y los usuarios al poseer un lenguaje común. Su ventaja sobre otros paradigmas radica en la propuesta de hacer software reutilizable y de fácil mantenimiento, para reducir así costos y tiempo de desarrollo. Para lograr esto se basa en la modularidad de sus sistemas que permite la elaboración de cada módulo por separado con una gran independencia entre sí que ayuda a la detección, aislamiento y corrección de errores si ellos se presentan.

Contar con software reutilizable y de fácil mantenimiento es una necesidad en esta época en donde el desarrollo del hardware esta avanzando a mayor velocidad que el del software impidiendo un máximo aprovechamiento de la nueva tecnología, por otro lado si se observa que el setenta y cinco por ciento (75%) del costo y esfuerzo de un sistema esta en el mantenimiento, una reducción en esta etapa ayudaría a realizar más proyectos a menor precio haciendo favorable la relación costo-beneficio.

La programación orientada a objetos es un estilo de programación que implementa cada una de dichas teorías para la solución de problemas computacionales. El ciclo de desarrollo de software orientado a objetos es muy sencillo (Fig. 1.), se tienen las etapas de análisis, diseño e implementación, a diferencia de los demás (el de cascada tradicional, por ejemplo) los cambios entre una y otra etapa son bastante pequeños permitiéndole más continuidad y menos pérdida en las transiciones,

adicionalmente, se aplican las teorías de prototipos al retroalimentar entre cada una de las etapas. Lo cual en conjunto, va a provee a los usuarios de un software que satisface fielmente sus necesidades porque el sistema final no sufre alteraciones por ser adaptado entre las etapas y se puede ir ajustando a lo largo de todo el ciclo de desarrollo a petición oportuna del usuario.

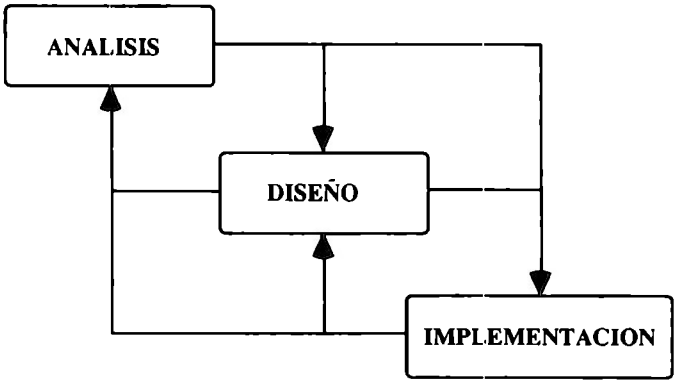


Fig. 1. Ciclo de desarrollo orientado a objetos.

En un aspecto un poco más técnico, soluciona el problema de los primeros métodos de programación (y algunos contemporáneos) en donde los datos corrían el riesgo de poder ser modificados o accesados desde cualquier parte del programa porque eran compartidos, ya que se construye un paquete con los datos y sus procedimientos para accesarlos. También se puede decir que es el resultado del proceso evolutivo de la abstracción de datos, logrando que los programas no sean orientados a los procedimientos sino a los datos, brindándoles así mayor seguridad e integridad en las aplicaciones.

4.2. Referencias

4.2.1. Universidades

Para conocer qué y cómo se esta impartiendo en la Ciudad de México la materia de Programación Orientada a Objetos, se hicieron averiguaciones en siete de las universidades situadas en esta zona, la Universidad Anahuac del Norte, el Instituto Tecnológico Autónomo de México (ITAM), la Universidad del Valle de México, la Universidad Iberoamericana (UIA), la Universidad del Nuevo Mundo (UNUM) y el Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Estado de México y Ciudad de México .

La tabla comparativa se encuentra en el Apéndice A-I y está organizada de la siguiente manera, en la parte izquierda aparecen cada uno de los conceptos considerados importantes del paradigma de objetos, siendo la base de la comparación; para evitar discrepancias la selección se hizo con base en la experiencia como profesor de la materia por tres semestres y la similitud a grandes rasgos presentada en los programas de estudio. Para cada universidad contemplada existe una columna en donde se coloca en cada uno de los conceptos el puesto que ocupa en la secuencia de temas que se sigue en esa universidad, si no se aparece el tema aparece una X. Al final de los conceptos se incluye una lista con los lenguajes de programación que se utilizan como soporte en cada sitio. Todo esto es con el fin de encontrar algún patrón o estándar a seguir.

Se puede apreciar que en el cincuenta y siete por ciento (57 %) de las universidades analizadas no se imparte la materia, esto es, que no la tienen

dentro de su currículum educativo, lo cual indica el poco énfasis que se le está dando a esta área de la computación en el país, siendo un tópico aconsejado por el comité educativo de la Asociación para la Maquinaria Computacional (ACM) en los programas de sistemas computacionales.

Es importante notar que no esta dentro de los objetivos de este trabajo juzgar a las universidades que no imparten la materia de programación orientada a objetos, sin embargo sí puede servir como plataforma para su implementación en futuros planes de estudio en esas instituciones.

De las tres universidades que imparten la materia, el ITAM, la UIA y el ITESM-CEM, sólo en este último se hace como una materia completa e independiente; en las otras dos, la parte fuerte se da como un subtema de la materia de diseño junto con otras metodologías de desarrollo. Como consecuencia y para mejorar un poco más este análisis, se incluyeron dos programas de estudios externos, uno propuesto por James McKim del Hartford Graduate Center (Apéndice A-II) y otro de la universidad estatal de Oklahoma (Apéndice A-III).

Como resultado del análisis comparativo de currícula, los conceptos básicos que deben ser contemplados dentro del paradigma de objetos son:

1. objeto
2. clase
3. método
4. instancia
5. mensaje
6. herencia y

7. polimorfismo

Sin embargo, no se conoce el significado que se le da en cada universidad a estos términos.

Es difícil determinar una metodología a seguir porque la gran mayoría agrupa varios conceptos bajo un mismo tema y se pierde el detalle de la secuencia de los programas. A pesar de todo, existe una similitud con la secuencia propuesta por el ITESM-CEM porque los temas que se dan son parecidos; en lo que más se difiere es en la forma de impartirla, ya que dos de ellos dividen los temas en dos materias a diferencia del ITESM-CEM que lo realiza en una sóla.

Situándose en la parte baja de la tabla: los lenguajes, el más utilizado es C++, aunque en la universidad estatal de Oklahoma no ofrecen el estudio de más lenguajes porque no los tienen según comentario del Dr. K.M. George - profesor asociado de las misma -.

4.2.2. Libros de texto

Una fuente casi natural de conocimiento son los libros de texto. Apoyándose en esto se extrajo información de trece referencias al tema que se comparan entre sí (Apéndice B), para saber con base en estas comparaciones cuáles son los conceptos más comunes - aquellos que aparecen en más del 70% de los libros - el resultado fue el siguiente:

1. objeto
2. diseño orientado a objetos
3. mensaje

4. encapsulamiento
5. clase
6. método
7. polimorfismo
8. herencia
9. herencia múltiple
10. clases abstractas
11. tipo abstracto de dato y
12. encubrimiento de información

Esto corrobora la primera apreciación basada en las universidades y adiciona algunos otros conceptos importantes que no habían sido tomados en cuenta.

En la columna de propuesta analizada de la tabla, se puede ver cuál es la secuencia más común para enseñar los conceptos fundamentales entre los libros tomados en cuenta. Se llegó a este resultado de la siguiente forma: en la columna de cada libro aparece un número por tema que corresponde al puesto que ocupa dicho tema en la secuencia que sigue el libro, luego se promediaron los lugares y dio el puesto promedio en la secuencia global, pero este resultado no se tomó tal cual, se extrajeron de ahí sólo los conceptos mencionados entre los que están en más del setenta por ciento (70%) de los libros, categorizados como importantes y sus respectivos puestos.

Estos resultados muestran que el primer concepto en ser enseñado debe ser el de objeto, le siguen en orden: encapsulamiento, tipo abstracto de dato, método, diseño orientado a objetos, mensaje, clase, herencia, polimorfismo, encubrimiento de información, clase abstracta y herencia múltiple. Mostrando una secuencia bastante lógica desde el punto de vista de las dependencias, ya que

cuando se necesita de algún concepto para definir o comprender otro, éste ya se ha definido.

En ninguno de los resultados se menciona una metodología de enseñanza aunque esta implícita en el orden que sigue el libro. De las definiciones se puede mencionar que es difícil determinar cuál o cuáles utilizar porque muchos textos están basados en algún lenguaje o ambiente de programación haciendo las definiciones muy específicas.

El lenguaje que más se utiliza es Smalltalk, lo cual es bastante lógico por ser el descendiente directo del pionero en el paradigma, Simula, y ser tal vez el más puro de todos; en ocurrencias de uso le sigue C++.

4.2.3. Encuestas

Para analizar la opinión del cliente (alumno) se aplicó una encuesta (Apéndice C-I) a treinta y seis (36) alumnos del ITESM-CEM, las condiciones para la selección de los alumnos fueron que estudiaran alguna de las carreras de sistemas computacionales y cursaran el quinto semestre o mayor, se deseaba que ya hubieran tomado la materia de POO pero no era requisito indispensable.

Los resultados se muestran en las tablas del Apéndice C-II. Los cinco alumnos encuestados que no han llevado la materia, se mostraron interesados en llevarla, confirmando la importancia adquirida últimamente por este tema.

Nótece que como parte de la encuesta hay un espacio en donde se da la oportunidad de proponer temas del paradigma de objetos adicionales a los vistos

en clase y dentro de los resultados no hay ninguna anotación en este punto, y además, de los temas cuestionados se considera unánimemente que todos deben ser vistos.

El orden sugerido por los resultados de la encuesta consiste en empezar por los conceptos más primitivos como objeto y mensaje, ver después conceptos más complejos como la herencia y llegar al análisis y diseño orientado a objetos. Esto con base en el orden en que ellos pondrían los temas. Se puede apreciar adicionalmente, que el dieciséis por ciento (16 %) opinó que lo que se debía dar primero era el análisis y el diseño, después los conceptos primitivos.

Los alumnos en sus propuestas para mejorar la clase opinan adicionalmente que requieren más teoría, esto no quiere decir que la prefieren sobre la práctica porque eso está contemplado en otro punto de la encuesta. A lo que se refieren es a no mostrar lenguajes sino explicar o analizar más los conceptos del paradigma. También sugieren profundizar más en el análisis y diseño aunque desean hacer más programas.

Entre los temas, el lenguaje que más se quiere ver es Eiffel (96 %) y le siguen en importancia C++ y Smalltalk, nótese que se prefiere ver primero C++ a Smalltalk. Por otro lado y más debido al auge de las interfases gráficas que a otra cosa, ya que son la nueva tendencia computacional y los alumnos desean mantener su ventaja competitiva, por la necesidad de crear aplicaciones interactivas y mucho más fáciles de usar, desean ver la interacción del lenguaje con las plataformas gráficas, adicionalmente se aprecia de cierta manera que hay interés por profundizar en los lenguajes, ya que entre los otros temas propuestos están los de aplicaciones prácticas e interacción entre lenguajes.

Siguiendo con los alumnos, el lenguaje que hubieran preferido ver y no vieron fue C++, mostrando que hay una gran preferencia por C++, debida posiblemente a que es mucho más comercial, y se sabe que lo van a poder utilizar casi sobre cualquier plataforma facilitándoseles su comprensión por su similitud con C.

Otro dato interesante que se puede obtener es que el ochenta y tres por ciento (83 %) de los alumnos que vieron un lenguaje desearía haber visto otro, pero de los que vieron tres lenguajes sólo el dieciséis por ciento (16 %) quisiera estudiar otro, se puede concluir que ver tres lenguajes de programación en un semestre es bastante carga, pero ver uno sólo es poco, el treinta y ocho por ciento (38 %) de los que pasaron por esto pidió menos lenguajes, aunque fueron pocos los que pidieron más teoría, indicando esto que se desea practicar en pocos lenguajes, mostrando también la preferencia de los alumnos por utilizar dos lenguajes.

4.2.4. Otros estudios

Para aprovechar lo que se ha hecho a este respecto, se exponen y evalúan trabajos de cuatro expertos en el tema. Se comenzará por el trabajo del señor James McKim[1] ([MCK 92] y [MCK 93]); en cuanto a la secuenciación que él propone, se puede decir que no tiene ningún fundamento teórico, por lo menos no lo incluye en su trabajo, al igual que la mayoría, tiene su base en la experiencia.

[1]James C. McKim, Jr. es profesor de la escuela de ingeniería y ciencias del Hartford Graduate Center.

Su curso se imparte por seis semanas y el resto de las clases se dejan para reuniones con los grupos de trabajo, para solucionar dudas y verificar avances del proyecto. En el ITESM-CEM esto no aplica ya que las clases hay que darlas en su totalidad presenciales. Además como requisito para el éxito de su clase los alumnos que ingresen deben ser buenos programadores.

Justifica por otro lado hacer el curso profundo y no quedarse en ver muchos conceptos por encima, para lograr así un convencimiento en los alumnos de las virtudes del paradigma. Esto es bastante bueno ya que da una idea de cómo se debe dar el curso según su experiencia.

En su trabajo al tocar el tema de los lenguajes, McKim toma como base tres: Eiffel, C++ y Smalltalk, de los cuales escoge Eiffel por su pureza y sencillez para ser el que se estudie a profundidad y sirva de soporte a todo el curso, no escoge C++ aunque es el más comercial, por ser muy complejo, diverso y demasiado compatible con C; a Smalltalk lo descarta porque no posee validación de tipos ni herencia múltiple principalmente.

Según McKim el proyecto es el alma del curso, después de una gran exposición concluye que los proyectos deben ser juegos porque contienen todo lo que requiere para un buen proyecto, cubre todos los temas del paradigma, son divertidos y tienen la complejidad adecuada. Coloca tareas cada dos semanas aumentando la complejidad entre una y otra.

Otro investigador, Thomas Wu[2] ([WU 93]) propone que se debe preparar al alumno para que pueda trabajar con cualquier lenguaje de programación y

[2]C. Thomas Wu es profesor asociado de ciencias computacionales en la Naval Postgraduate School.

metodología de desarrollo y aclara que el decir programación orientada a objetos no elimina el enseñar metodologías de análisis y diseño. Como método de enseñanza se indica que inicia con la comprensión del lenguaje sobre el que se va a trabajar y gradualmente va incorporando nociones de objetos hasta completar todo el paradigma, lo llama "bottom-up".

Del lenguaje opina que lo importante es que no contradiga el paradigma y debe ser híbrido para facilitar un aprendizaje incremental de los conceptos porque a partir de lo que se conoce se van construyendo elementos de objetos más complejos, y recomienda Prograph. Lo más interesante de su disertación es que menciona que es tan malo programar mal en lenguajes puros como, medio programar en objetos con lenguajes híbridos, cuando analiza las ventajas o desventajas de usar uno u otro tipo de lenguaje.

D'Souza[3] ([DSO 93]) es otro experto que ha escrito acerca de cómo enseñar programación orientada a objetos, en su exposición se muestra la ventaja del ciclo de desarrollo de software orientado a objetos comparado con el tradicional. También se menciona la utilidad de incluir en dicho ciclo la elaboración de sistemas por prototipos. Sugiere adicionalmente que es más importante enseñar los fundamentos del diseño que los detalles de la programación o el lenguaje.

No aconseja ni Smalltalk ni C++, como lenguajes de programación para la enseñanza, el primero lo descarta por su complejidad debida principalmente al gran número de clases que tiene, ya que el curso se confundiría en su

[3]Desmond D'Souza es el fundador de ICON Computing Inc. una empresa especializada en educación y tecnología orientada a objetos. Es autor de varios artículos y cursos sobre análisis, modelaje y diseño orientado a objetos.

BIBLIOTECA
64776

comprensión y navegación; C++ no se recomienda, aunque es el que más se utiliza, porque es muy similar a C y se pierden esfuerzos al hacer cambiar la forma de pensar del alumno; él lo utiliza sin incluir muchos elementos como sobrecarga de operadores y herencia múltiple, que son fundamentales en el paradigma.

El último autor que se presenta en esta parte es Meyer[4] ([MEY 93]), cuya idea consiste en dar la materia con el método de apertura progresiva de cajas negras, en donde a partir de un ejemplo con unas pocas líneas de código y reusando varias librerías, el alumno se va adentrando en el lenguaje y los conceptos, profundizando en cada uno de los elementos utilizados, de lo general a lo específico; agrega de cierta forma un modelo basado en prototipos porque a partir de esa primera construcción se hacen otras más complejas.

La opinión de Meyer acerca del lenguaje ideal no se considera porque él es el creador de Eiffel y su opinión a este respecto podría estar muy sesgada, aunque su lenguaje es pequeño y orientado a la educación. Su presentación del proyecto es ambiciosa, al igual que el curso, debe durar todo un año y como si fuera poco debe permitir que al siguiente periodo se trabaje sobre él, una aproximación a la creación de software reutilizable.

4.2.5. Psicología del aprendizaje

Dejando a un lado a los conocedores de objetos y atendiendo ahora al proceso de aprendizaje humano, hay que notar que en este ambiente, aprender a programar bajo el paradigma de objetos, no corresponde a un simple aprendizaje

[4]Bertrand Meyer es presidente de Interactive Software Engineering, Inc. y editor de Prentice Hall en la serie de libros orientados a objetos. Es uno de los creadores del lenguaje Eiffel.

por asociación en ninguno de sus tipos condicional, por ensayo y error, y secuenciación, como pasa casi siempre, es mucho más complejo que eso.

Tomando como referencia a Gagne ([GAG 85]) y partiendo de que lo importante para conocer el aprendizaje es enfocarse en lo que se aprende, en nuestro caso, la programación crea habilidades que pueden considerarse dentro de la capacidad de estrategia cognitiva, porque el individuo controla su forma de aprender, recordar y pensar para analizar los problemas y crear los programas que los solucionen, lo cual cumple con lo que se entiende por estrategia cognitiva. De igual forma, al programar se tienen que poder distinguir y clasificar tanto las situaciones o elementos del problema como aquellos que van a servir para representarlos, formando así un soporte para el aprendizaje de la programación que se puede considerar una habilidad intelectual.

Siendo lo anterior cierto, se debe cumplir como condición interna para su aprendizaje el que haya aprendido con anterioridad las bases para poderlo hacer, algo sobre lo cual se pueda actuar, en programación orientada a objetos esto es, los conceptos fundamentales del paradigma y la estructura de los lenguajes que le van a permitir formar las reglas de la programación; como condición externa sólo se recomienda práctica con situaciones nuevas para usar diferentes métodos o programas, lo cual ayuda en un futuro a recordar las habilidades que vayan necesitando a cada momento.

Si se analiza un poco más a fondo la parte de la condición interna para el aprendizaje de la programación se puede apreciar que los conceptos son el soporte de la formación de las reglas que más adelante nos van a ayudar a solucionar problemas, y el conocimiento del lenguaje nos ayuda a armar cada

vez estructuras más complejas a partir de unas simples para construir la solución general del problema.

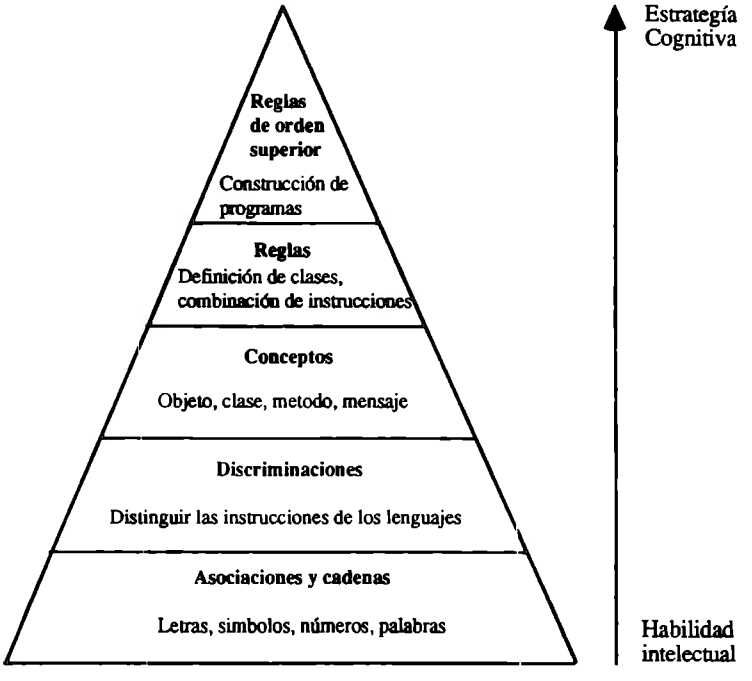


Fig. 2. Capacidades aprendidas.

La Fig. 2 muestra como a partir de los conocimientos básicos adquiridos por asociación como las letras y los números, se puede ir construyendo la solución de un problema con un programa. Con los símbolos y números formamos las palabras reservadas de los lenguajes de programación y luego distinguimos una de otra por su función, también se van formando los conceptos fundamentales tanto del paradigma - en este caso - como del lenguaje utilizado

para poder combinarlos a partir de sus reglas y formar el programa para solucionar el problema. Adicionalmente se aprecia la evolución de las habilidades adquiridas por las personas clasificadas como intelectuales hacia las de estrategia cognitiva.

4.3. Selección y definición de los conceptos fundamentales

Tradicionalmente los conceptos tienen una gran importancia en la elaboración de programas de estudio, antes de determinar y definir cuáles son los conceptos fundamentales del paradigma de objetos, es bueno dar dos ideas que justifican el iniciar el trabajo por este punto, la primera es de un experto en objetos y educación: D'Souza ([DSO 93]) que dice que "El contenido de la educación se basa en los conceptos"; la otra es la obtenida de Gagne ([GAG 85]) en donde expone, como elemento primario para la creación y aprendizaje de reglas o relaciones al concepto, que a su vez forma parte de las habilidades intelectuales sobre la que se basa el proceso de aprendizaje de la programación.

Conceptos del paradigma de objetos hay muchos, la selección de los más representativos se hizo con base en lo expuesto en la sección anterior, de donde se destaca como lo más útil los resultados del análisis de los libros de texto ya que incluyen los resultados del análisis de las otras universidades, los programas de estudio y encuestas realizadas a los alumnos. Para que sea totalmente adecuado se deben hacer algunas adaptaciones, entre los conceptos más comunes no aparece el de las relaciones de uso, sin las cuales no se cumpliría en gran medida la reutilizabilidad del código bajo el paradigma de objetos; estas relaciones se manejan como tal en pocos libros pero son muy reducidos los ejemplos en donde no se incluyen. Otra adaptación es incluir el concepto de análisis orientado a objetos, para que se cubra todo el ciclo de desarrollo,

no tiene mucho sentido el que se pueda mencionar el diseño sin haber pasado por el análisis.

Se concluye que los conceptos fundamentales a tratarse en la programación orientada a objetos son:

1. objeto
2. diseño orientado a objetos
3. mensaje
4. encapsulamiento
5. clase
6. método
7. polimorfismo
8. herencia
9. herencia múltiple
10. clases abstractas
11. tipo abstracto de dato
12. encubrimiento de información
13. análisis orientado a objetos y
14. relaciones de uso

No todos estos conceptos se encuentran en el mismo nivel, unos son más abstractos que otros como el caso del encubrimiento de información. Dentro de los conceptos anteriores hay también unos que conforman otros como método y clase, y otros que para poder existir necesitan de algunos como polimorfismo y herencia.

Ahora bien, para la definición adecuada de cada uno de los conceptos, se hace una distinción, si se selecciona la de alguno de los libros que son escritos teniendo

como base un lenguaje de programación como Eiffel u Objective C en los casos de Meyer ([MEY 88] y [MEY 92]) y Cox ([COX 91]) respectivamente, la definición se realiza acorde con ese lenguaje. Por ejemplo, una definición aceptable de objeto en Eiffel o C++ es: variable a la cual se le puede enviar algún mensaje; esto no aplicaría en Smalltalk ya que las clases en este lenguaje también pueden recibir mensajes.

La aclaración anterior deja como alternativa seleccionar la definición de los conceptos entre los autores de libros enfocados al paradigma puramente y al análisis o al diseño orientado a objetos. Por lo que es conveniente tener más de una definición ya que entre sí se complementan o a algunos alumnos se les facilita entender más unas que otras, además, como se verá más adelante, cuando sea la parte del tutorial, vamos a necesitar por lo menos dos.

La definición más básica y en la que se basa el paradigma de POO es la de objeto. Además sobre ella recae el nombre del paradigma. Un objeto es algo que tiene sentido en el ambiente de la aplicación, también se puede decir que es la unión de los datos y sus operaciones - y desde un punto de vista más técnico es, un paquete de software que contiene una colección de datos y procedimientos relacionados con ellos -, en general es una entidad que existe y que tiene un comportamiento bien definido, sobre la cual guardamos datos y operaciones para manejarlos. Es importante notar que el objeto tiene una identidad, esto es, se puede distinguir uno de los demás. A los datos de los objetos también se les llama atributo o variable.

Existen otras definiciones ambiguas que sugieren ser recursivas y dan la apariencia de sinónimos; por ejemplo la que dice que un objeto y una instancia son lo mismo. En el resto de este trabajo se manejarán como sinónimos.

Bajo condiciones normales cuando se refieren a un objeto se hace sobre algún elemento que se conoce o se puede imaginar, volviéndose un poco detallista se puede deducir que un objeto es miembro de alguna clase, por ejemplo el triángulo de la Fig. 3 es de la clase de triángulos rectángulos. Entonces definimos una clase como la descripción de un conjunto de objetos que tienen una estructura (datos) y comportamiento común.

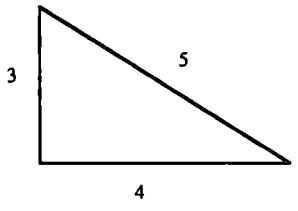


Fig. 3. Triángulo.

Se ha mencionado varias veces el término operación y comportamiento. Bajo este paradigma se llamará a la descripción de estos dos términos como método siendo éste la forma en como se manipulan los datos; es la especificación de los pasos que se siguen en cada una de las operaciones. Para muchos es el equivalente a los procedimientos de otros paradigmas.

Un mensaje es la petición que se le hace a un objeto para que ejecute una acción o se comporte de alguna manera, desde el punto de vista funcional, es la operación que se hace sobre un objeto. La recepción de un mensaje conduce a la ejecución de un método.

Los cuatro anteriores conceptos son entre todos los aquí mencionados los más importantes del paradigma, sin la comprensión de ellos el resto no tendría un buen

fundamento, por eso y para no quedarse con la sola definición es bueno poder determinar las relaciones existentes entre cada uno de ellos.

Ha sido una tendencia natural del ser humano la clasificación de los elementos que tiene a su alrededor, como son los animales, minerales, vegetales, símbolos y gente dentro de las empresas. Lo que se hace cuando se clasifica es que se ordena de cierta manera el conocimiento, se puede hacer una analogía entre el proceso de creación de un sistema bajo el paradigma de objetos y la clasificación, en ambos se clasifican (u ordenan) los elementos en las clases que van a formar parte del ambiente. Una clase pueden ser los animales, que se pueden a su vez subdividir en la clase de los vertebrados y los invertebrados, a cada una de estas se le pueden asociar objetos que puede ser cualquier mascota o animal que se conozca. Las clases sirven para clasificar a los objetos.

Un objeto no puede existir sin una clase a la cual pertenece, y cuando un objeto es de alguna clase se dice que es del tipo de la clase, esto es, el tipo del objeto es la clase a la que pertenece. En las clases está el formato general de los métodos y atributos que se van a incluir en cada uno de los objetos miembros de esa clase.

Cuando ocurre un mensaje el objeto que lo recibe ejecuta el método con el mismo nombre de la operación que viene asociado con el mensaje. La clase debe tener un método con el nombre de la operación que se le va a aplicar al objeto para no presentar errores.

El siguiente concepto es polémico, debido a que algunos autores asumen que el encapsulamiento y el encubrimiento de información son similares. Se considera en este estudio que son diferentes y con base en eso se trabajará. Encapsulamiento es la

capacidad de crear bloques o paquetes, lo que se ha venido dando desde las unidades (Units) en Pascal. Lo que interesa más en este punto es el encubrimiento de información que es la capacidad de esconder aquellos detalles de la implementación, en este caso de las clases, que no son relevantes para su comprensión, básicamente la forma como hacen lo que se les pide.

El concepto de tipo abstracto de dato (TAD) es anterior cronológicamente al de objeto y es uno de los que más influyó en el proceso evolutivo hacia los objetos, es un tipo de dato definido por el usuario que consiste en los datos y los algoritmos para manejarlos con el ingrediente adicional del encubrimiento de información. Aparentemente es igual a lo que hasta el momento se ha manejado como objeto, pero en las siguientes líneas se mostrará la diferencia.

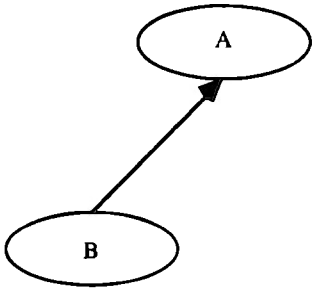


Fig. 4. Relación de herencia, B hereda de A.

La herencia es el mecanismo más poderoso del paradigma de objetos, sirve para definir clases especializadas de una más general, en donde autornáticamente se incluyen los atributos y métodos de la clase general en la especializada. A través de ella se pueden compartir (pasar) las estructuras entre las clases que así lo requieran. Esto se debe a que algunas clases están organizadas jerárquicamente y ésta es una forma de

relacionarse. Si una clase B hereda de una clase A se representa gráficamente con una flecha (Fig. 4.), cuando esto pasa en B se tendrá lo mismo que en A, más las características que la hacen única.

En este punto es aconsejable distinguir dos tipos de clases, la *subclase* que es la que hereda de alguna otra y la *superclase* que es de la cual heredan. Cuando una clase posee más de una superclase, esto es, hereda de más de una clase, se dice que tiene una herencia múltiple (Fig. 5.). En este punto se puede hacer la distinción en cuanto a los métodos que existen para evitar los conflictos que se presentan cuando se hereda lo mismo de dos clases diferentes, a saber, renombrar una de las dos o calificarlas con el nombre de la clase a la cual pertenece el que se va a utilizar.

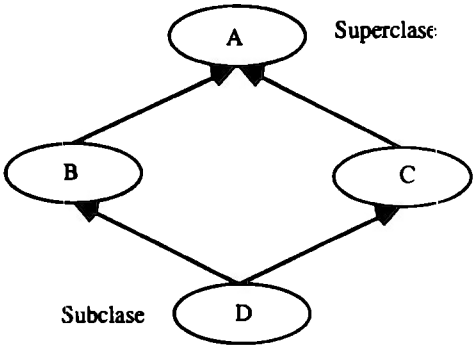


Fig. 5. Subclases, superclases y herencia múltiple.

El concepto que hace la diferencia entre la programación con TAD y la orientada a objetos es la herencia, los TAD's no manejan herencia, para algunos POO = TAD + herencia.

Otra manera que tienen las clases para relacionarse es partir de las relaciones de uso, las cuales se dan a través de la incorporación de objetos de otras clases como atributos de una tercera, indicándonos la existencia de objetos complejos construidos con base en otros, también son conocidas como relaciones *parte-de*. Junto con la herencia son las formas más comunes de relaciones entre clases. Las relaciones de uso se pueden representar con una flecha doble (Fig. 6.).

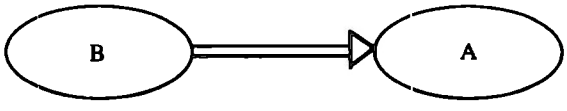


Fig. 6. Relación de uso, la clase B usa a la clase A.

Un calificativo que pueden recibir las clases es el ser abstractas o virtuales. Se entiende por clase abstracta aquella clase muy general que puede no tener objetos, cuya utilidad radica en poder definir con ellas las estructuras básicas de otras clases. Realmente no deberían tener instancias pero hay lenguajes de programación que permiten definir objetos de ese tipo.

Gracias a la herencia se genera un nuevo concepto, el polimorfismo. Éste se puede manejar de dos maneras, dependiendo si se toma como referencia el objeto o el método, si descomponemos la palabra en sus raíces latinas tendríamos "muchas formas", basándonos en el objeto nos indica que un nombre de objeto podría referirse a objetos diferentes (de diferentes clases), si nos basamos en los métodos tendríamos un mismo nombre de operación con varias implementaciones.

En este último caso se puede decir que se tienen varias alternativas de operaciones, se aplica la adecuada dependiendo del tipo del objeto. A este proceso de búsqueda de la implementación adecuada se le llama encadenamiento dinámico.

Frecuentemente la definición dependiente de los métodos se asocia a la sobrecarga de operadores y lo que pasa es que éste término en particular sólo aplica a métodos llamados como las operaciones tradicionales de suma (+), resta (-), multiplicación (*) y división (/).

Faltan dos definiciones esenciales: análisis y diseño orientados a objetos. Aquí, más que definiciones, se necesitan realmente distinguir qué es lo que se debe hacer en cada una de estas etapas de desarrollo. Para ubicarlas, el ciclo de desarrollo orientado a objetos es bastante particular. Consiste básicamente de análisis, diseño e implementación con la ayuda de los prototipos. Como el curso es de programación y además existen muchísimas metodologías orientadas a objetos que además se ven en otras materias del currículum Desarrollo de Sistemas II y Sistemas VI, sólo se profundizará en la tercera etapa, las demás se verán muy sencillamente.

El análisis orientado a objetos (AOO) consiste como en todo análisis, de la comprensión del sistema y la especificación de los requerimientos, el toque personal del paradigma por decirlo así, radica en que eso se hace detectando cuáles son las clases y objetos que ayudan a modelar el problema.

En el diseño orientado a objetos (DOO) además de pulir y completar las clases se adicionan las relaciones entre ellas para crear las jerarquías y mecanismos que logren el comportamiento deseado del sistema.

Hasta este momento se tienen los catorce conceptos más importantes del paradigma de objetos, con las definiciones más generales para no dejar por fuera alguna interpretación. Cabe señalar que las definiciones se encuentran todas al mismo nivel para facilitar después su incorporación a la metodología o secuenciación instruccional.

4.4. Metodología de enseñanza

La metodología de enseñanza es el alma del curso, ya que de ella depende el que se aprendan o no los conceptos o temas que se imparten. A continuación se hará un análisis de la situación de la metodología de la enseñanza de la programación orientada a objetos y se adaptará alguna de las mencionadas en secciones anteriores o se propondrá la ideal.

Como punto de partida y a lo que se refiere específicamente al campo del aprendizaje, se hace notar el hecho de que algunos de los programas de estudio existentes o sugeridos se hayan realizado con base en la experiencia de los que imparten la materia, no indica que sea un trabajo mal hecho, la verdad es que la técnica más común - aún entre los psicólogos - para determinar un programa de estudio adecuado es la validación del mismo por un grupo de expertos en el tema. Por lo que todas las metodologías mencionadas en las secciones anteriores son igualmente válidas.

Para filtrar los análisis que se han mencionado hasta el momento y algunas apreciaciones futuras, hay que destacar que la metodología que aquí se propone tiene como límite el estar planeada para una materia con una intensidad de aproximadamente cuarenta y dos (42) horas de clases presenciales, con una frecuencia de tres (3) horas a la semana. Y para cumplir con los requisitos del Instituto, según el reglamento

académico, cada cuatro semanas se aplicará una evaluación a los alumnos para conocer su situación, y al final del curso se debe presentar un proyecto final.

En una evaluación sobre cuál de las alternativas expuestas en la sección 4.2 es la ideal, o con cuál de los programas el alumno aprende más y aplica efectivamente sus nuevos conocimientos en desarrollos futuros, se aprecia que la información en los estudios presentados por McKim, D´Souza, Wu o Meyer no tienen una muestra de los resultados obtenidos por los alumnos al final del curso, ni una comparación contra alguna otra metodología. Gracias a esto se considera que la mejor forma de enseñanza es una mezcla de las ideas más relevantes de cada una, incluyendo los análisis de los libros de texto, universidades, encuestas y teorías de aprendizaje.

Con esta premisa y analizando las propuestas descritas anteriormente se concluye lo siguiente, hay que aprovechar la continuidad del ciclo de desarrollo orientado a objetos que es parte importante de la ventaja de este paradigma sobre otros; ver durante el curso por lo menos una metodología de análisis y diseño ya que varios autores hacen énfasis en que la enseñanza de la programación debe contener más dedicación a los principios del diseño que a los detalles de los lenguajes, y también haciéndole caso al Dr. J.C. Esteva[5] que comenta: "No tengas el mismo problema de todos, sin metodología", refiriéndose a que la mayoría de los profesores cuando imparten la materia de programación orientada a objetos no incluyen dentro de sus temarios una metodología de desarrollo. Los alumnos deben ser buenos programando para no tener que explicar el lenguaje desde sus raíces, y por último basarse en los prototipos para permitir un aprendizaje incremental. Siguiendo así los consejos de Meyer y D´Souza y atendiendo la petición de los alumnos en cuanto a ver análisis y diseño.

[5]Juan Carlos Esteva es doctor en inteligencia artificial y profesor asociado de Eastern Michigan University.

La metodología propuesta es la siguiente: el conocimiento del paradigma a partir de su ciclo de desarrollo. Con esto se va a dividir el curso en tres partes. La primera, que trata el análisis orientado a objetos, le sigue la de diseño orientado a objetos y la última etapa es la de programación o implementación. La misma metodología ayuda al ordenamiento de los conceptos, ya que se hace de acuerdo al momento en que se van necesitando, dándole a la metodología de desarrollo la importancia que requiere.

Lo primero que se debe hacer antes de iniciar con el análisis y sus definiciones es motivar a los alumnos con la materia diciéndoles qué es lo que se espera de ellos y el por qué del paradigma. También es recomendable mencionar la evolución hacia los objetos, el paso de la programación lineal a la modular y de objetos pasando por el manejo de procedimientos; la diferencia de orientarse a los datos y no a los procedimientos. Con esto se satisface una condición de aprendizaje indicada por Gagne acerca de las bases que se deben tener para adquirir un nuevo conocimiento, ya que de esta forma se estandarizan conocimientos y se nivela el punto de partida.

Dentro de cada etapa, ya sea análisis o diseño, se impartirán los temas correspondientes a cada una de ellas en el orden obtenido por el análisis de los libros de texto básicamente ya que el orden de los temas resultante de los demás análisis - universidades y encuestas - no está acorde con la secuenciación instruccional obtenida en la sección de psicología del aprendizaje o se cubre con el resultado de los libros.

Con la primera parte del curso se procede de la siguiente forma. Para hacer un buen análisis de acuerdo a su definición, se necesita saber lo que es una clase, un objeto, sus partes, relaciones y comportamiento, con lo cual cubriríamos los temas de objeto, encubrimiento de información y encapsulamiento, tipo abstracto de dato, método, mensaje y clase. Al mismo tiempo se les muestra con ejemplos en un lenguaje

de programación cada uno de ellos. En este punto se hace una modificación, si se sigue fielmente el resultado de los libros de texto, en vez de encubrimiento de información se debe ver encapsulamiento, pero por ser considerados por muchos autores como sinónimos se manejan al mismo tiempo, como si fuera uno sólo (Fig. 7.).

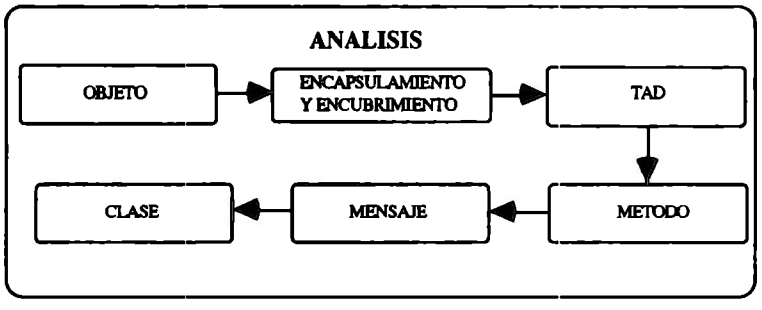


Fig. 7. Metodología del análisis.

Como metodología de análisis se propone una sencilla sugerida por Pressman ([PRE 88]) en donde la selección de los elementos resultantes del análisis se hace con base en un párrafo o texto escrito con la descripción del problema de la siguiente forma: a partir de los sujetos se obtienen los objetos, con los verbos se obtienen los métodos, de los adverbios los parámetros de los mensajes y a partir de los adjetivos se logran los atributos del objeto.

Después con el diseño orientado a objetos, y para hacer lo que corresponde a esta etapa, se necesita de la comprensión de los conceptos de relaciones de uso, herencia, polimorfismo, clase abstracta y herencia múltiple (Fig. 8.). Con todo eso logramos el objetivo del diseño: relacionar las clases y objetos para lograr un modelo

del sistema que incluye su comportamiento. De la misma forma que en el paso anterior, se dan ejemplos con un lenguaje, pero con la novedad de que se parte de lo que se hizo antes. En los resultados se obtienen las relaciones de uso como el último concepto a ser tratado pero es más natural formar objetos complejos con las relaciones de uso que aprovechar lo que ya esta hecho con la herencia.

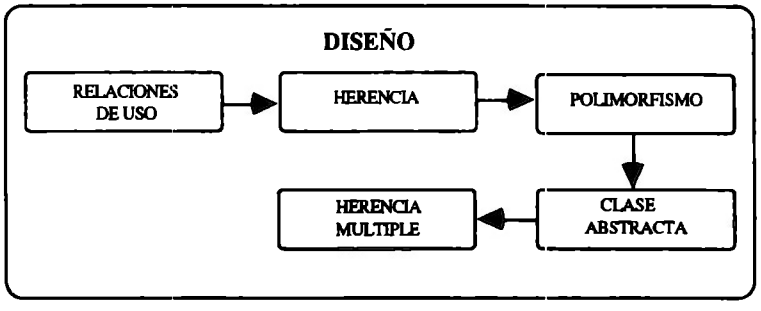


Fig. 8. Metodología del diseño.

Es conveniente y para seguir reforzando las metodologías de desarrollo, la enseñanza de unos diagramas llamados de clases sugeridos por Meyer (Fig. 4. y Fig. 6.) que facilitan la comprensión del diseño.

Hasta aquí, lo que se tiene es el aprovechamiento del ciclo de desarrollo orientado a objetos. Se parte de clases generales como análisis para ir a lo específico como el concepto y su representación en un lenguaje, se trabaja con prototipos para hacer construcciones a medida que el ciclo lo pide con base en lo que se tiene, se respeta el orden preestablecido por los libros, y no se deja de incorporar una metodología. Cabe notar que la secuencia instruccional no tiene problema porque los

conceptos que conforman otros se ven antes ayudando a la comprensión del más complejo.

Con lo anterior adicionalmente se respeta el proceso de aprendizaje explicado en Gagne en donde se deben tener unas bases de conocimientos que faciliten el aprendizaje de los conceptos, en este caso vistos durante la etapa de análisis, para después formar reglas en el diseño que le darán más forma al paradigma. En lo que resta se mostrará cómo estando en la etapa de implementación se satisface lo de la práctica gracias a la programación, cubriendo todas las condiciones de aprendizaje analizadas en la sección 4.2.5.

Esto toma aproximadamente veinte (20) horas, para dejar otro tanto igual al estudio de la programación, donde se verán a fondo dos lenguajes partiendo con el precedente de la comprensión del paradigma. Como consideración adicional se debe tener el no incluir especificaciones formales porque los alumnos no tienen las bases matemáticas para dominarlos.

En la metodología se debe tener también en cuenta el lenguaje que se va a utilizar y servir de apoyo para la comprensión del paradigma. Los llamados lenguajes orientados a objetos (LOO) son muchos y se pueden clasificar primordialmente en dos categorías, los puros y los híbridos. La diferencia fundamental radica en que en los primeros sólo se puede programar bajo el paradigma de objetos, los segundos son extensiones de otros lenguajes - de tercera generación casi todos - por lo cual permiten programar fuera del paradigma de una manera tradicional. De los lenguajes que aquí se tratarán tenemos de los considerados puros, Eiffel y Smalltalk, de los híbridos C++ y Prograph, este último por ser más de programación visual que de objetos. Existe otra

clasificación, aunque menos común, y consiste en clasificarlos en orientados a objetos o basados en objetos, dependiendo de si permiten o no la herencia.

Como se puede percibir en los análisis de la sección 4.2., las opiniones son muy diversas en cuanto al lenguaje, lo que unos argumentan como ventaja para utilizar uno para otros es una desventaja y de acuerdo a su punto de vista cada uno siente tener la razón. Lo que sí es notable es la popularidad de C++ y la complejidad de Smalltalk.

En este trabajo no se puede contemplar el utilizar Prograph. No es por el lenguaje en sí, ya que éste es muy bueno y poderoso. Un problema que se le encuentra es su complejidad, no es fácil navegar a través de todos sus elementos. Otro problema que tiene es el sistema operativo sobre el que corre, Macintosh, ya que es poco común el desarrollo de sistemas en él.

Enfocándose en la situación de nuestros alumnos en la sala de computadoras, se dispone de cuarenta (40) Macintosh que la mayoría del tiempo están siendo utilizadas para procesamiento de texto, así que no tendrían oportunidad de trabajar a placer sobre ellas. Adicionalmente, varios de ellos cuentan con una computadora IBM o compatible en la casa.

Smalltalk es de los lenguajes más complejos que hay, cuenta con aproximadamente dos mil (2000) clases definidas y el ambiente gráfico sobre el que trabaja la versión que se tiene (Visual Works ver. 1.0) le cambia a los alumnos la forma de apreciar un programa pero les facilita el desarrollo de interfases gráficas, además, como todo se hace a partir de lo que está hecho se aprecia claramente la importancia de la reutilizabilidad. Es interpretado lo cual evita los procesos de compilación y encadenamiento ofreciendo un ambiente siempre actualizado.

C++ es un lenguaje híbrido basado en C, es bastante comercial y no es difícil encontrar un compilador del mismo en cualquier plataforma, es el más práctico de los que se están analizando y se consigue de acuerdo a las necesidades de cada quien. Eiffel es el menos popular y no les sirve de mucho a los alumnos conocerlo a fondo cuando salen a trabajar, sin embargo es el de más fácil aprendizaje ya que es un lenguaje pequeño con una sintaxis sencilla, adicionalmente permite la creación de programas ejecutables y la implementación de todos los conceptos del paradigma de objetos.

Lo que se recomienda durante la primera mitad del periodo académico es utilizar ejemplos en Eiffel al mismo tiempo que la impartición de los conceptos, ya que es bastante sencillo y ayuda a la comprensión de los conceptos fundamentales del paradigma por su pureza, su sintaxis parecida a Pascal los obliga a pensar en objetos pero agiliza la comprensión, además su compilación y manejo de archivos es similar a lo que los alumnos están acostumbrados, el objetivo no es dominar el lenguaje ni conocerlo todo, es sólo utilizarlo como soporte.

Después de dominar los conceptos se pasa a ver Smalltalk, pero este lenguaje sí se estudia a profundidad y se aprovecha la facilidad de crear aplicaciones gráficas. Este proceso durará un mes, ya que no debe haber problema con asimilar los conceptos, se dedicarán a implementarlos con este lenguaje. Su poder les hará apreciar las maravillas del paradigma, podrán utilizar muchas clases ya definidas y obtener buenos resultados rápidamente.

El último lenguaje por estudiar es C++, ya en este punto se garantiza que el alumno no tiene problemas con el paradigma entonces no se corre el riesgo de una

tendencia a programar tradicionalmente y se puede llegar a detalles específicos del lenguaje, llamados a ambientes gráficos particularmente. Con esto el alumno siente que va a tener más utilidad lo que esta aprendiendo y que es real.

De ejemplos es poco lo que se encuentra en la literatura. Para este fin la experiencia será la base, tomando como ayuda algunos detalles del proceso de aprendizaje. Los ejemplos que mejores resultados han dado son aquellos que les muestran cosas familiares a los alumnos, por lo general son sencillos y caen en un dominio por todos conocido.

Como el curso está basado en el ciclo de desarrollo y prototipos, lo ideal es trabajar con sistemas que faciliten su desarrollo progresivo como por ejemplo el representar un sistema universitario, que va desde la definición de las personas que se encuentran en él hasta las dependencias entre las diferentes áreas ya sean administrativas o docentes. De manera independiente puede haber ejemplos que soporten algunos conceptos como la representación de las figuras geométricas o los fraccionarios.

Entre los ejemplos y por no utilizar sólo código en algún lenguaje de programación, es bastante aconsejable incluir gráficas para que los alumnos puedan asociar las definiciones a figuras lo cual les va a permitir memorizar fácilmente lo que están aprendiendo. Un ejemplo de esto puede ser el dibujo que se encuentra en la página siguiente (Fig. 9) utilizado por Taylor ([TAY 90]), que ayuda a entender la diferencia entre clase y objeto, y cómo una clase se puede ver como una fábrica de objetos.

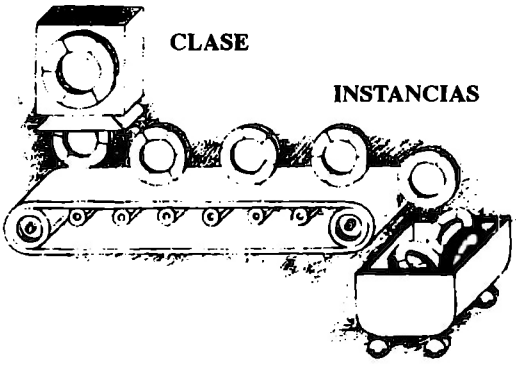


Fig. 9. Clases e instancias [TAY 90].

A los alumnos les gusta más la práctica que la teoría, así que se deben colocar ejercicios para realizarse dentro y fuera de la clase. En la clase es bueno comenzar por darles todo el material del ejercicio (su solución) y a medida que transcurre el semestre ir quitándoles la tutela dejándolos más libres para que realicen sus propias soluciones. Se puede aprovechar de los primeros casos y aplicarlos a cada uno de los lenguajes. Si se desea justificar el hecho de dar más casos prácticos por el solo gusto de los alumnos, está demostrado experimentalmente que ejemplos prácticos coadyuvan a la comprensión y transferencia de aprendizaje.

La idea es que realicen tareas para que demuestren la comprensión de cada uno de los conceptos y como proyecto final un sistema más completo, el cual deben definir, para ser realizado a través de todo el semestre con eso les permite cubrir todo el ciclo de desarrollo. Un juego es recomendable porque es bueno hacer algo que tanto uno como los demás utilicen con gusto, lo malo es que el manejo de la interfaz no es tan simple como para empezar por ahí, se podrían asignar juegos pequeños y sencillos

como tareas. Tanto los ejercicios como el proyecto final se hacen en grupos máximo de tres (3) personas, de esta forma tienen mayor colaboración y retroalimentación entre ellos, es ademas una buena oportunidad para aprender a trabajar grupalmente.

5. Tutorial

5.1. Aplicación de la alta tecnología en la educación

A medida que pasan los años la tecnología ha provisto a todos de elementos cada vez más poderosos, que permiten implementar cosas que antes no se podían siquiera imaginar, por eso es bueno poder evaluar en este punto si hay algo que pueda servir para hacer tutoriales mejores que los actuales y más eficientes que las clases tradicionales. Por enseñanza tradicional se entenderá aquella que ocurre en un salón de clase sin ningún otro recurso audiovisual que el pizarrón.

Un buen punto de partida para saber en qué medida ayuda la tecnología a la educación es la determinación de los medios adecuados para el aprendizaje de algún tópico que se desee. En este caso el tópico que se desea enseñar es la programación orientada a objetos y, por sus características, se puede considerar como una habilidad intelectual y no motora, ya que es la creación de una solución computacional a un problema a través de un proceso racional, no manual. Lo cual está acorde con lo que se mencionó en secciones anteriores porque las habilidades intelectuales y las estrategias cognitivas, que son el marco del aprendizaje de este paradigma, se pueden agrupar bajo la característica común de ser una habilidad mental.

Observando el diagrama de flujo (Apéndice D) propuesto por Gagne ([GAG 85]) para seleccionar los medios ideales para facilitar el aprendizaje de acuerdo al tipo de habilidad que se desee enseñar, se notan como medios sugeridos para la enseñanza de una habilidad mental la computadora, el texto programado y la televisión interactiva cuando las personas son alfabetas - saben leer y escribir - y autodidactas - aprendan solos, sin un profesor al lado -. Analizando un poco el resultado y la situación actual de

la tecnología se puede concluir que todo esto se tiene bajo un concepto conocido como multimedia.

Sería aconsejable entonces entender un poco de qué se trata esto de multimedia. Como su nombre lo indica son muchos medios, pero muchos medios de transmisión - en este caso de conocimiento - integrados a través de una computadora, con un valor agregado, la interacción con el usuario. Entre los medios más comunes se encuentran: el texto, las gráficas, el sonido y el video que cubren la sugerencia de Gagne.

El texto es un medio de expresión considerado de banda baja, esto es, se transmite relativamente poca información en un segundo (300 bits/seg aprox.), las gráficas, el sonido y el video son medios de banda alta, transmiten del orden de los millones de bits por segundo. A simple vista esto no dice mucho pero es realmente importante como se podrá ver enseguida.

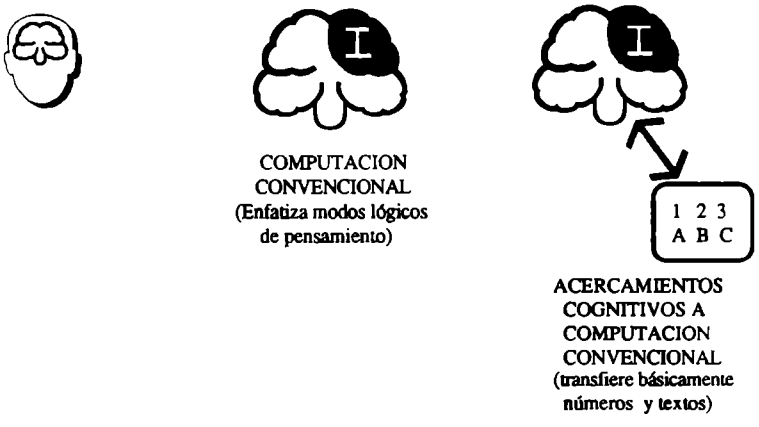


Fig. 10. Computación y enseñanza tradicional. Fuente [WOD 93].

Tradicionalmente, en particular en los primeros tutoriales, se utilizaban las computadoras con ambientes textuales - sólo caracteres, letras - y este caso, al igual que la enseñanza convencional, ejercitan el lado izquierdo del cerebro (Fig. 10) que está especializado, por decirlo así, en el procesamiento de números y texto, lo cual implica, siendo un poco drásticos, desperdiciar la mitad del cerebro de los alumnos en este caso.

Con la multimedia al tener a todos los medios interactuando entre sí y con el usuario se va a ejercitar también el lado derecho del cerebro, el del procesamiento de imágenes, teniendo así mayor utilización del mismo (Fig. 11). De esta forma la impresión del usuario llega a ser mejor y le va a facilitar el pasar a un nivel de conciencia superior, que es en la que se almacena la comprensión del ser humano.

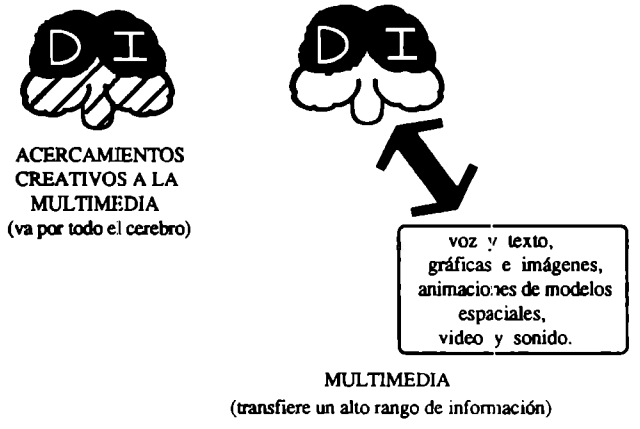


Fig. 11. Multimedia y enseñanza. Fuente [WOD 93].

El solo hecho de tener los medios no es suficiente, hay que utilizarlos y combinarlos bien, se le debe permitir al usuario interactuar con ellos de la forma más adecuada.

Utilizar multimedia tiene otras ventajas sobre la forma tradicional de enseñar, a continuación se listan las más importantes:

- Es más accesible. Rompe limitantes de tiempo y espacio
- Permite ambientes personales (privados) de trabajo, sin juicios
- Permite a cada alumno avanzar a su propio ritmo
- Se adapta a los diferentes estilos de aprendizaje
- Es activa
- Posee un amplio canal de transmisión

Algunas de las características anteriores son debidas al solo hecho de ser tutorial como el permitir ambientes personales en donde el alumno no va a tener que preocuparse si va al mismo ritmo de los demás o preguntar algo que cause que lo intimiden. Otras como el ancho del canal de transmisión de información sólo se aplica a los tutoriales realizados con multimedia.

No todos los alumnos aprenden igual (culturas y capacidades diferentes) y al estar en un salón de clase casi siempre se deja de considerar esto y se imparten las clases como si todos fueran iguales, la clase es la misma para todos los presentes, es menos personalizada la educación. Además cada uno tiene su propio ritmo de comprensión y avance, con tutoriales en multimedia se superan ambos problemas.

En un estudio realizado por IBM ([BIS 90]) se muestra cómo es tres veces más efectiva la enseñanza con video interactivo que con la enseñanza tradicional. Otro

estudio, pero éste sobre niños, muestra como tienen una actitud positiva a aprender con la ayuda de las computadoras. Se concluyó que sí es más efectivo este método que el tradicional y que los mantiene más concentrados. Extrapolando estos dos resultados y uniéndoles las ventajas de la multimedia se puede ofrecer un producto que cumple con importantes factores para el éxito del tutorial con multimedia.

Hay también quienes argumentan que la tecnología necesaria para implementar este tipo de tutoriales es muy costosa. Pero es, como en todo, un asunto de costo-beneficio, qué tanto se le va a incluir cuando se calcula el monto, si el costo del equipo, el personal que trabaje sobre él, el software, la capacitación o algún otro. Por otro lado, lo que casi siempre se olvida es el beneficio intangible que puede traer consigo como en todos los sistemas.

Entre las consideraciones hay que incluir que el tiempo de preparación de una clase con multimedia es tres o cuatro veces mayor que el de una clase normal ([HAZ 92]). Por eso es aconsejable tanto para ambientes o temas estables - perdurables - lo cual no es factible en el caso de la programación si ésta se quiere fijar en algún lenguaje porque estos evolucionan. Es mucho mejor para la enseñanza de los conceptos fundamentales ya que siempre serán los mismos o sufrirán sólo pequeñas variaciones.

Si la tecnología educativa funciona o no en última instancia es un factor que depende de muchas cosas, entre otras la factibilidad del proyecto, la confiabilidad del software y hardware que se utilizan, la relación costo-beneficio y el impacto que ésta cause. Lo que sí es seguro es que a través de los tiempos ha cambiado la forma de adquirir el conocimiento, se ha vuelto más sofisticada la forma de aprendizaje ([KEA 93]).

En general sirve y aporta bastante la tecnología en la educación, pero para un óptimo resultado de los tutoriales en multimedia, se necesita de soporte técnico (para mantener el equipo en buenas condiciones), la colaboración de los compañeros (otros profesores) y el apoyo de la administración (la dirección).

5.2. Características de un buen tutorial

El tutorial es una herramienta y para su uso y efectivo resultado se necesita algo más que sólo tenerlo (Fig. 12.). Si se analiza el gráfico de abajo, se puede apreciar que se necesita alguna técnica (en este caso el uso del computador) para poder manejar el tutorial. También se va a necesitar algún método - opciones de los menús, secuenciación - para trabajar con él adecuadamente en orden y por último, un objetivo que motiva o incita a su utilización. Con esto se puede apreciar el nexo entre la mente y los medios que se utilizan para actuar sobre ella, conformando toda la cadena de lo que es una interfaz.

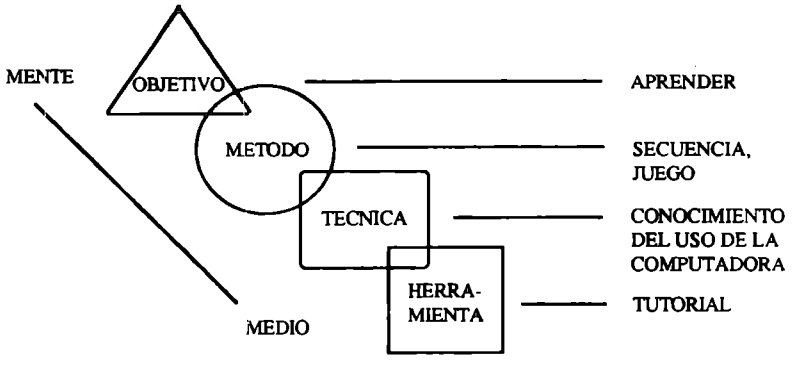


Fig. 12. Adaptación niveles genéricos.

Para efectos de este trabajo, el alumno ya posee la técnica para manejar una computadora con sus respectivos dispositivos, ya sean mouse o teclado. En cuanto al método, éste fue determinado en el capítulo anterior de esta investigación y se proveen otros dos que se explicarán mas adelante. El objetivo de este tutorial en particular debe ser para los alumnos el aprender el paradigma de objetos sin implicar la eliminación de las clases magistrales, al contrario utilizarlo como un asesor permanente del curso. Lo que falta por estudiar son las características de la herramienta, que se tratan a continuación.

Existe una metodología para hacer tutoriales y esta expuesta en [CRA 90] pero no será utilizada por considerarse muy enfocada a ambientes textuales y no gráficos que como se mostró son más aconsejables por lo que no aplica en este caso.

Antes de comenzar a describir características o pensar en cualquier cosa hay que tener en cuenta que lo más importante de un tutorial es la metodología o programa de estudio que lo soporte, no importa que tan novedosa y poderosa sea la tecnología, ni tampoco sirve que sean muy bonitas las pantallas, si no hay un fundamento detrás de todo.

Los tutoriales deben ser divertidos para que el alumno lo utilice por gusto, entonces deben tener:

- Objetivos claros
- Retroalimentación inmediata
- Sensación de control sobre las acciones
- Representar desafío, que requiera de habilidad
- Una oportunidad de terminar
- La oportunidad para concentrarse

- Inmersión
- Absorción de tiempo sin que se sienta

Roger C. Schank ([SCH 93]) menciona que un tutorial debe ser activo y no sólo debe contener botones. Si es pasivo la gente se aburre, pues no los hace partícipes del proceso, siendo activo se sienten partícipes del mismo; si sólo contiene botones se pierde el interés, se vuelve pura selección. Un producto de este tipo debe ser motivante, no simple, esto es, con un nivel no muy difícil o avanzado como para que cause estrés, o tan sencillo que se aburran con él. También debe contener varias respuestas y acceso a una especie de base de datos de donde escoger lo que se desee y cómo se desee para darle al usuario la posibilidad de que él controle su aprendizaje y lo haga a su medida. El usuario debe poder navegar por todas partes.

Por su parte, Sunil Hazari ([HAZ 92]) argumenta que las presentaciones de multimedia deben ser fáciles de utilizar e interesantes, en esto se incluyen tanto los medios a través de los cuales se hace, como los diseños implementados. También menciona que debe ser flexible, esto es, permitirle al usuario avanzar a su propio ritmo y ajustarlo a sus gustos, lo que concuerda con lo expuesto hasta el momento.

Otras de las características principales de un tutorial son: el poderse amoldar a la capacidad de aprender del usuario, y que sea completamente factible el que le ayude a incrementar sus conocimientos. También se debe tomar en cuenta que una de las bases del aprendizaje son las asociaciones verbales de lo que se ve y escucha, esto indica que las imágenes que se utilicen deben ser escogidas con sumo cuidado, ya que de ellas depende la primera impresión del usuario que después va a formar parte de su reconocimiento.

Un tutorial se puede hacer de acuerdo a un modelo del profesor, esto es, a la habilidad que se tenga para distribuir el conocimiento, pero en estos casos se depende mucho del material, que se desea enseñar. Para seguir esta teoría, lo más aconsejable es que posea un nivel físico, uno lógico y una vista, en otras palabras un almacenamiento y equipo, una conceptualización y una forma de verlo, muy similar a una base de datos, permitiéndole ver lo mismo (los conceptos) de diferentes formas. También se pueden hacer tutoriales de otras dos formas, con base en el aprendiz o en el contenido pero en estos casos es más complicado. De lo anterior se puede rescatar el hecho de poder ver lo mismo en mas de una forma.

Se menciona en [KUB 93] la importancia de la secuenciación pero no aplica directamente en este caso porque la idea es que el usuario determine lo que quiere hacer. Lo que sí se puede rescatar de ese escrito es el hecho de colocar en varias partes del tutorial algunos puntos de interés para no perder la concentración del alumno.

Por otro lado y considerando que es un sistema computacional, el tutorial debe ser accesible, consistente y acorde al presupuesto ([SHN 92]). Concluyendo, un tutorial debe ser:

- Divertido
- Con una metodología que lo soporte
- Activo
- Controlable por el usuario
- Interesante y desafiante
- Flexible
- Un facilitador del aprendizaje
- Con varias vistas
- Accesible (disponible)

- Consistente

Que el usuario pueda hacer lo que desee no quiere decir que su seguimiento va a ser desordenado, una vez que se seleccione lo que se desea se va a seguir un orden o secuencia de partes.

5.3. Selección de hardware y software

Siendo un tutorial que va a ser utilizado por alumnos en el ITESM-CEM, lo más recomendable es comenzar por saber con qué se cuenta en el Instituto tanto en hardware como en software. Ya que se reducirían los costos utilizando lo que hay y se incrementaría la accesibilidad del equipo.

En el ITESM-CEM se cuenta con el siguiente equipo computacional:

- 43 Macintosh II Vx
- 30 SUN
- 24 PS/2 modelo 30
- 80 RS/6000
- 14 SNAKE

Recientemente se adquirieron unas IBMultimedia con el fin de ser utilizadas para desarrollos y aplicaciones de multimedia pero no pudieron ser consideradas en esta investigación porque no se tenían al momento en que se inició.

Por su cantidad, la conclusión directa sería implementar el tutorial sobre las RS/6000, ya que los alumnos tendrían mayor posibilidad de accesarlo. Sin embargo, sobre estas máquinas no hay un software especializado para trabajar con multimedia.

La opción a considerarse de acuerdo a la cantidad de máquinas existentes pasó a ser la de las Macintosh, y adicionalmente sirve para aprovechar que estas computadoras cuentan con dispositivos extras que la convierten en una buena alternativa. Poseen por ejemplo monitores de miles de colores y unidad para discos compactos.

En cuanto al software, si se toman en cuenta las bondades del paradigma de objetos, lo ideal es desarrollar el tutorial con programación orientada a objetos, desafortunadamente no existe un lenguaje de POO que permita la manipulación de video y sonido indispensables en un tutorial con multimedia.

Teniendo lo anterior como premisa y analizando lo concerniente al software, se tomó en cuenta que para trabajar con multimedia se necesita de alguna manera poder manipular imágenes, texto, video y sonido independientemente para después poderlos integrar. El software existente en el Instituto sobre la Macintosh de acuerdo a su utilidad es:

- Para texto e imágenes:

  MacDraw, SuperPaint, Painter, Photoshop e Infini-D.
- Para video:

  Premier y Video Shop.
- Para sonido:

  SoundEdit.
- Para integrar:

  MacroMind Director.

Del software para texto e imágenes por su gran facilidad para el manejo de colores y su diversidad de herramientas se utiliza Painter, Photoshop e Infini-D, aunque el primero se trata de evitar porque crea archivos demasiado grandes - más de un megabyte del disco duro -. En cuanto a la paquetería para manipulación de video se utilizó el Video Shop para la digitalización de las películas y el Premier para la edición.

5.4. Diseño e implementación

La elaboración del prototipo del tutorial incluye contemplar las características encontradas en este trabajo acerca de lo que debía contener un tutorial y reglas de diseño de interfases. Entre las reglas fundamentales del diseño se encuentra que los botones que se utilicen deben ser grandes, la pantalla no debe contener más de cuatro colores y sus contrastes deben ser adecuados, el sistema debe ser consistente y el sonido no debe perturbar, pero sí llamar la atención. Con base en esto se comenzó por el diseño en papel de cada uno de los elementos que se iba a tener y luego su elaboración en el paquete adecuado.

Lo primero que se hizo fue darle un nombre al tutorial, se escogió DAO siglas de Diviértete y Aprende Objetos por representar lo que se desea de este tutorial, ademas casi todas las siglas involucradas en el paradigma de objetos tienen tres letras DOO, AOO y POO entre otras, dándole así más consistencia.

La metodología que soporta el tutorial fue determinada en el capítulo cuatro, sin embargo para no caer siempre en la misma secuencia, se deben crear dos opciones adicionales para que cuando el usuario entre al tutorial escoja la de su agrado. Una vez invocado el tutorial, el alumno podrá escoger por medio de amplios botones entre seguir la metodología de la clase, ir de acuerdo al índice de cada uno de los conceptos o

entrar en un juego, dándole así la oportunidad de avanzar de acuerdo a su estilo y divertirse. Estos tres botones conforman la pantalla principal del tutorial (Fig. 13.).

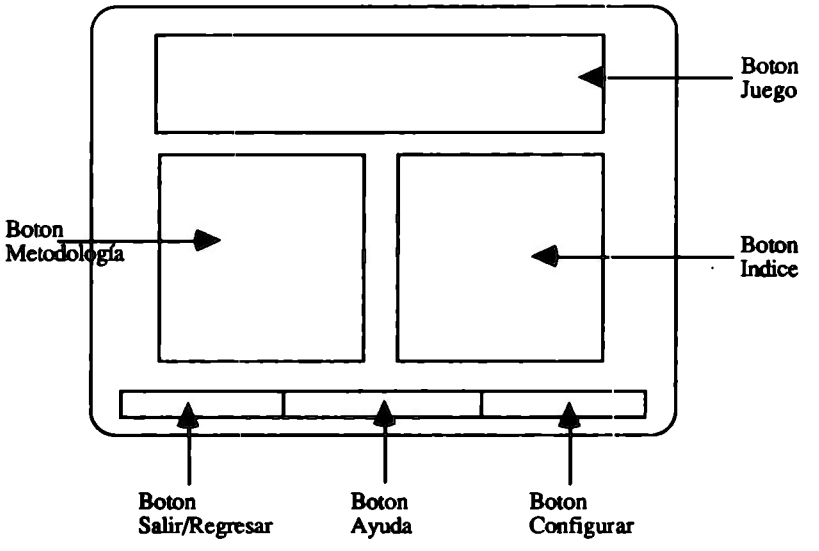


Fig. 13. Pantalla principal.

La pantalla principal al igual que todas las demás pantallas debe contener en la parte inferior otros tres botones que le permitan al usuario desde cualquier parte del tutorial configurarlo a su gusto, regresar al menú principal u obtener ayuda acerca de su uso. Configurarlo implicaría escoger el sexo de la persona que le gustaría le hable (hombre o mujer), la música que escuchará de fondo y el lenguaje de programación que debe aparecer primero dentro de los ejemplos. Regresar al menú anterior lo devuelve a la pantalla principal y si se encuentra en este punto, lo envía al sistema operativo. La ayuda es del uso general del sistema. Esta parte es constante a través de todo el sistema y es algo que le da consistencia al tutorial.

Cuando el alumno oprima el botón del ciclo de vida del software el sistema le mostrará en la pantalla básica de enseñanza los conceptos del paradigma de objetos según el orden de la metodología especificada en el capítulo cuatro. Si el alumno oprime el botón de índice, le va a aparecer una pantalla con un botón por cada concepto fundamental detectado en el capítulo anterior, brindándole la posibilidad de ver el que se desee, el cual será mostrado en la pantalla básica de enseñanza. El tercer botón es el de juego, que como su nombre lo dice llama al juego del tutorial.

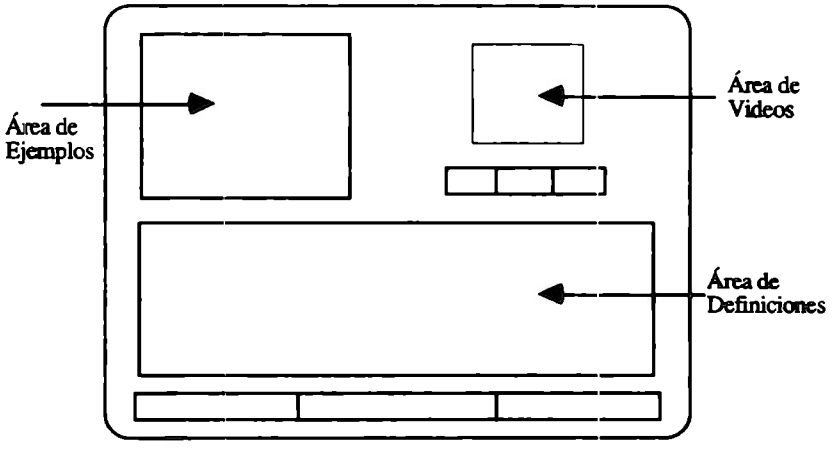


Fig.14. Pantalla principal de enseñanza.

La pantalla principal de enseñanza para cada concepto está dividida en tres partes, en la primera se va a tener un ejemplo explicando el concepto en los lenguajes orientados a objetos: Eiffel, C++ y Smalltalk, permitiendo verlos todos, en otra parte va a haber un video explicatorio el cual puede ejecutarse o no y en la otra parte se podrá apreciar un hipertexto con las definiciones del concepto que está tratando (Fig. 14.).

Cuando se selecciona alguna de las palabras subrayadas del hipertexto aparecerá una ventana con una explicación del significado de ésta.

En el juego se van a tener dos etapas: en la primera una serie de ahorcados que van a proporcionar las palabras que a su vez serán utilizadas en la segunda etapa para completar un crucigrama sin números, para cada palabra del ahorcado se van a tener tres oportunidades para completarse. Adicionalmente debe contener una especie de banco de preguntas que se seleccionen de manera aleatoria, para evitar que memoricen las preguntas y la monotonía.

Las pantallas se diseñaron con colores que no cansan al usuario para que éste pueda estar largo tiempo si lo desea, los botones son amplios y con íconos autoexplicativos para que no tenga ningún problema con su uso. Los colores y diseños son iguales a través de todo el tutorial.

La implementación tuvo algunos limitantes, como la imposibilidad de utilizar más de dos canales de sonido o una base de datos para preguntas. Sin embargo el modelo se llevó a cabo. Lo primero que se implementó fueron las pantallas en Photoshop, Painter e Infini-D, luego se hizo la filmación de las películas a incluirse, después se grabó el sonido y por último se integró todo en MacroMind Director.

6. Discusión

Algo que no se mencionó pero sí hay que tener presente, es la situación de las instalaciones. Si se desea en un futuro implantar la utilización de tutoriales realizados con multimedia hay que adaptar las instalaciones, se deben tener máquinas rápidas, disponibles, con buenos monitores y todo el equipo necesario ya sea discos compactos, video discos o equipo de sonido. También hay que planear a largo plazo su elaboración y no perder la idea de que un tutorial con multimedia consume cuatro veces más tiempo en su elaboración que uno de clase normal ([HAZ 92]).

Una primera mejora que se le podría hacer al tutorial, es el incorporarle algunas características de inteligencia artificial, convertirlo en un sistema inteligente para así aumentar la interacción con el usuario a través de diálogo en lenguaje natural; de esta misma forma se le puede también brindar al usuario la posibilidad de practicar la programación, lo cual es importante.

En cuanto a la metodología, lo más aconsejable es hacer un seguimiento a su implantación en el semestre 9308. Lo más recomendable es utilizarla en un grupo piloto para poder comparar resultados, pero se considera complicado de realizar ya que el plan de estudios de la misma materia en diferentes grupos debe ser igual por reglamento, y a su vez, para evitar también problemas con los alumnos.

Por otro lado y no menos importante, es la elaboración de todo el tutorial; en este punto hay que reevaluar y adquirir el software y hardware necesario porque el que se tiene no permite realizar todo lo que se plantea. También se puede contemplar la posibilidad de que una vez que se termine se pueda grabar en un disco compacto para su fácil distribución.

Otra mejora que se puede hacer es en el juego, crear uno más activo en donde no se den sólo letras, donde existan caminos y animaciones para que el usuario se motive más. O si no, varios juegos de acuerdo al nivel de experiencia y conocimiento del alumno. Es general, realizar juegos de acuerdo a diferentes tipos de aprendizaje.

La idea final debe ser el incluir al tutorial dentro de una herramienta de programación orientada a objetos, para completar así todo un ambiente de aprendizaje e implementación del paradigma.

7. Conclusiones

Las conclusiones se pueden dividir en tres partes, la primera para tratar la metodología, la segunda con las consideraciones del tutorial y la tercera acerca de una comparación entre los resultados aquí obtenidos y la experiencia de haber impartido la materia por tres semestres.

En cuanto a la metodología es bueno mencionar que las definiciones y la secuencia de los temas son las partes que más difícilmente cambiarían, ya que las definiciones son básicamente siempre las mismas, se trata de los conceptos fundamentales y estos no cambian, y el orden de los contenidos está muy bien soportado aunque puede cambiar de acuerdo a los resultados obtenidos con su implantación este semestre. Las partes que más área de oportunidad representan son, los ejemplos y proyectos, ya que de ellos se podría dar un conjunto para cada tema, y los lenguajes de programación porque evolucionan con el tiempo.

Se puede planear la utilización de la versión tres de Eiffel ya que adicionalmente contiene una muy buena interfaz gráfica, o utilizar el Little Smalltalk que permite interpretar cada uno de los conceptos del paradigma de objetos. Adicionalmente se puede contemplar la necesidad de incorporar a mediano plazo lenguajes que facilitan el manejo de interfases gráficas sobre otras plataformas como Visual C++. Vale la pena mencionar que incluir un nuevo lenguaje no significa saturar las clases con un mayor número de ellos, la idea es que se reemplace alguno por otro mejor o actualizado de acuerdo a su tipo o papel desempeñado en la metodología aquí propuesta, esto es, si el objetivo de Eiffel es apoyar la parte docente que se sustituya por uno que optimice dicho objetivo y así para todos los lenguajes según su propósito.

Para el desarrollo del tutorial hay que considerar los resultados de la experiencia en la creación del prototipo del mismo. La elaboración de cada una de las pantallas en Photoshop puede ocupar hasta un megabyte (1 MB) de memoria, dependiendo de la resolución que se desee, la música por otro lado ocupa aproximadamente dos megabytes (2 MB) de memoria por cada minuto veinte segundos (1'20") de sonido y así tanto el video como la conexión entre todos los elementos van a formar un paquete demasiado grande, el solo modelo ocupa quince megabytes (15 MB) de memoria aproximadamente y es casi una décima parte de todo el trabajo dejando una requisición de memoria bastante considerable para un programa.

En el caso particular del ITESM-CEM, se cuenta con los recursos humanos tanto para la elaboración del tutorial como para llevar a cabo la metodología en un salón de clases. Sin embargo, para la realización de un tutorial con todas las características aquí expuestas, no se cuenta con el hardware o software necesario, este no es sólo un problema de este Instituto, sino de cualquier parte en donde se quiera implementar algo así, es uno de los principales problemas de la multimedia, consume muchos recursos y todavía no están todas las instalaciones adecuadas para eso. Por otro lado, hacerlo con algunas restricciones es algo bastante viable, y en este momento se cuenta con una nueva alternativa: las IBMultimedia.

Aprovechando los tres semestres de experiencia impartiendo la materia de POO se puede hacer una comparación con lo obtenido en este trabajo. Hay que destacar que ninguna de las veces se ha impartido la materia de la misma forma, esto es, con la misma metodología, la primera vez se utilizó sólo el lenguaje Eiffel y hasta después del primer mes, luego de un fuerte énfasis en la parte formal con TAD's, se comenzó a hablar de los conceptos del paradigma de objetos como tal; el segundo semestre se dieron los conceptos fundamentales del paradigma de objetos desde el principio para ubicar más a los alumnos y como lenguaje se utilizó adicionalmente C++; y la última vez se estudiaron tres lenguajes de programación: Smalltalk, Eiffel y C++. Los resultados y comentarios obtenidos en estos

cursos se ven fuertemente reflejados en la encuesta aplicada a los alumnos, por consiguiente todas las medidas tomadas con base en ella sirven para mejorar las clases de semestres anteriores.

La parte en donde más complicaciones han tenido los alumnos es en la abstracción de los problemas. Partiendo del objetivo final que es desarrollar un sistema y con base en su ciclo de desarrollo, se soluciona en gran medida dicha situación porque desde el principio saben a donde deben llegar. Otro punto bastante polémico ha sido el lenguaje que se les ha enseñado, no había un balance entre la cantidad y la utilidad del mismo, con esta propuesta se logra ubicar cada lenguaje en lo que es más eficiente, Eiffel en educación y formalidad por su poder de expresión, pequeño número de palabras reservadas y manejo de excepciones; Smalltalk en pureza y ambiente que obligan al usuario a pensar y manipular objetos a través de utilerías gráficas que facilitan su trabajo; y C++ en la diversidad de plataformas donde corre y el uso generalizado en el comercio e industria.

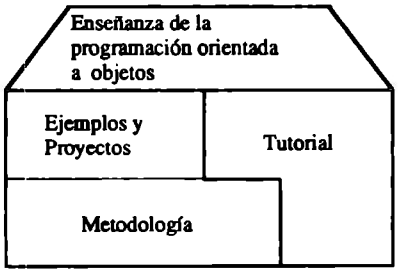


Fig. 15. Representación gráfica.

Generalizando este trabajo, se puede visualizar como un estudio que le sirve a los profesores de Programación Orientada a Objetos de cualquier institución para enseñar de una manera óptima su materia. Gráficamente se vería como en la figura 15, una metodología que

sirve de soporte tanto a la enseñanza de la programación orientada a objetos en un salón de clases como a través de un tutorial.

Para terminar, este trabajo no debe ser visto solamente como una forma de enseñar objetos en clase o con un tutorial, las implicaciones que tiene van más alla que eso. Los desarrolladores de software tienen la necesidad de producir aplicaciones a menor costo y con menor tiempo de elaboración para ir acorde a la tendencia de la época, la forma más adecuada para lograrlo es con la POO y si los conceptos no se tienen sólidos esto no se va a poder aplicar. Además si la experiencia durante su aprendizaje no es buena muy posiblemente no apliquen sus conocimientos en esta área. Como si fuera poco, la forma de conceptualizar y programar en objetos le da a los alumnos otra forma de ver el mundo ya que hasta este momento sólo conocen una vista estructurada del mismo, la de objetos es más natural.

La comprensión de los conceptos fundamentales del paradigma de objetos es la base tanto de futuros aprendizajes en el ramo como metodologías específicas de análisis y diseño, y programación en algunos lenguajes; como la adecuada utilización de los mismos, por eso hay que garantizar su correcto aprendizaje.

Bibliografía

[ASK 92] Askar, Petek et al. "Students' perceptions of computer assisted instruction environment and their attitudes towards computer assisted learning". Educational research, 34, 2, (verano 1992), 133-139.

[BAR 93] Bar-David, Tsvi. "Object-oriented education and training in the 1990's". Journal of object oriented programming, 6, 1 (marzo-abril 1993), 24-31.

[BAS 91] Bass, Len y Joelle Coutaz. Developing software for the user interface. U.S.A.: Addison Wesley, 1991.

[BIS 90] Bissonnet, Daniel. "The role of high technology in training". The technological horizons in education journal, 17, 10 (junio 1990), 51-53.

[BOO 91] Booch, Grady. Object oriented design with aplications. U.S.A.: The Benjamin/Cummings Publishing Company, 1991.

[BUD 91] Budd, Timothy. An introduction to object-oriented programming. U.S.A.: Addison Wesley, 1991.

[CAR 91] Cardoso, Rodrigo. Verificación y desarrollo de programas. Colombia: Presencia Ltda.,1991.

[COA 91] Coad, Peter y Edward Yourdon. Object oriented design. U.S.A.: Prentice Hall, 1991.

[COA 91a] Coad, Peter. "Why use object-oriented development?. A management perspective". Journal of object oriented programming, 4, 6 (octubre 1991).

[COM 93] Comite de Educación de la ACM. "ACM model high school computer science curriculum". Communications of the ACM, 36, 5 (mayo 1993), 87-90.

[COX 91] Cox, Brad J. y Andrew J. Novobilski. Object-oriented programming: An evolutionary approach. 2da ed. U.S.A.: Addison Wesley, 1991.

[CHE 91] Chen, Z. "From student model to teacher model: Enriching our view of the impact of computers on society". Computers & society, 21, 4 (octubre 1991), 46-48.

[DSO 93] D'Souza, Desmond. "An educated look at education". Journal of object oriented programming, 6, 1 (marzo-abril 1993), 40-46.

[ELL 90] Ellis, Margaret A. y Bjarne Stroustrup. The annotated C++ reference manual. U.S.A.: Addison-Wesley, 1990.

[GAG 85] Gagné, R.M. Las condiciones del aprendizaje. 4ta ed. México: Nueva Editorial Interamericana, 1985.

[HAZ 92] Hazari, Sunil. "Multimedia: Is it the right tool for your instructional application?". Journal of educational multimedia and hypermedia, 1, 2 (1992), 143-146.

[JOH 93] Johnson, Craig W. and Patricia A. Grover. "Hypertutor therapy for interactive instruction". Educational technology, 1 (enero 1993), 5-13.

[KEA 93] Kearsley, Greg. "Educational technology: Does it work?". Ed-Tech review, 1, 1 (primavera/verano 1993), 34-36.

[KHO 90] Khoshafian, Setrag y Razmik Abnous. Object orientation: concepts, languajes, databases, user interfaces. U.S.A.: John Wiley & Sons, 1990.

[KUB 93] Kuboni, Olabisi. "Teachers' treatment of content in the design of instructional materials". Educational technology, 1(enero 1993), 38-43.

[LAL 90] LaLonde, Wilf R. y John R. Pugh. Inside Smalltalk. Volume I. U.S.A.: Prentice Hall, 1990.

[MAR 92] Martin, James y James J. Odell. Object-oriented analysis & design. U.S.A.: Prentice Hall, 1992.

[MCK 92] McKim, James. "Teaching object-oriented programming and design". OOPSLA '92, Vancouver, B.C.

[MCK 93] McKim, James. "Teaching object-oriented programming and design". Journal of object oriented programming, 6, 1 (marzo-abril 1993), 32-39.

[MEY 92] Meyer, Bertrand. Eiffel: The language. U.S.A.: Prentice Hall, 1992.

[MEY 88] Meyer, Bertrand. Object-oriented software construction. Great Britain: Prentice Hall, 1988.

[MEY 93] Meyer, Bertrand. "Toward an object-oriented curriculum". Journal of object oriented programming, 6, 2 (mayo 1993), 76-81.

[MEY 71] Meyer, Susan M. Instrucción programada: Análisis de cuadros buenos y malos. México: Editorial Limusa-Wiley, 1971.

[PRE 88] Pressman, Roger S. Ingeniería del software: Un enfoque práctico. 2da ed. México: McGraw Hill, 1988.

[RUM 91] Rumbaugh, James et al. Object-oriented modeling and design. U.S.A.: Prentice Hall, 1991.

[SCH 93] Schank, Roger C. "Learning Via Multimedia Computers". Communications of the ACM, 36, 5 (mayo 1993), 54-56.

[SHN 92] Shneiderman, Ben. Designing the user interface. U.S.A.: Addison Wesley,1992.

[TAY 90] Taylor, David A. Object-oriented technology: A managers guide. U.S.A.: Addison Wesley, 1990.

[TUR 88] Turbo Pascal 5.5: Object-oriented programming guide. U.S.A.: Borland International, 1988.

[WIR 90] Wirfs-Brock, Rebecca et al. Designing object-oriented software. U.S.A.: Prentice Hall, 1990.

[WOD 93] Wodtke, Mark von. Mind over media. U.S.A.: McGraw-Hill, 1993.

[WU 93] Wu, C. Thomas. "Teaching OOP to beginners". Journal of object oriented programming, 6, 1 (marzo-abril 1993), 47-50.

[ZOR 93] Zorfass, Judith. "Curriculum: A critical factor in technology integration." The computing teacher, (febrero 1993), 14-15.

Apéndices

Apéndice A

Programas universitarios

Apéndice A-I

Tabla comparativa

	U. DEL VALLE DE MEXICO	ITESM CCM	ITESM CEM	OKLAHOMA STATE U.	UNUM
DISEÑO	X	X	8	3	X
OBJETO	X	X	1	1	X
CLASE	X	X	2	2	X
METODO	X	X	3	2	X
INSTANCIA	X	X	4	2	X
MENSAJE	X	X	5	2	X
HERENCIA	X	X	6	4	X
POLIMORFISMO	X	X	7	5	X
C++	X	X	3	1	X
PASCAL OO	X	X	X	X	X
SMALLTALK	X	X	1	X	X
EIFFEL	X	X	2	X	X

ITESM : Instituto Tecnológico y de Estudios Superiores de Monterrey
CCM : Campus Ciudad de México
CEM : Campus Estado de México
UNUM : Universidad del Nuevo Mundo

	U. ANAHUAC DEL NORTE	ITAM*	UIA*	JAMES McKIM Jr.
DISEÑO	X			1
OBJETO	X	1	1	2
CLASE	X	2	1	2
METODO	X	3	1	2
INSTANCIA	X	4	1	2
MENSAJE	X	5	1	2
HERENCIA	X	6	1	3
POLIMORFISMO	X	7	1	3
C++	X	1	X	1
PASCAL OO	X	X	1	X
SMALLTALK	X	2	X	1
EIFFEL	X	X	X	1

ITAM : Instituto Tecnológico Autónomo de México
UIA : Universidad IberoAmericana
X : No se da el tema
* : El análisis y el diseño O-O lo ven una materia antes

NOTA : Los números indican el orden en que los temas se imparten en cada institución

Apéndice A-II

Programa de James McKim

James C. McKim Jr. HGC

Syllabus

Week	Topic
1.	Introduction, Software Quality and Modularity
2.	Reusability and OO Design
3.	Objects and Classes
4.	Programming by Contract and Invariants
5.	Genericity and Exception Handling
6.	Inheritance, Dynamic Binding and Polymorphism
7.	Individual Meetings with Design Teams
8.	**MidTerm Exam**
9.	More on Inheritance and Polymorphism
10.	Designing Class Interfaces
11.	Constants and Shared Objects
12.	Memory Management
13.	Other OO Languages and Environments
14.	Catch-up, Review for Final
15.	**Final Exam**

Prerequisites: Programming experience at the level of *Systems Programming*

Apéndice A-III

Programa de la Universidad Estatal de Oklahoma

Course Title: Object-Oriented Software Construction

Text materials :

1. Timothy Budd, An Introduction To Object-Oriented Programming, Addison-Wesley, 1991.
2. Margaret A. Ellis & Bjarne Stroustrup, The Annotated C++ Reference Manual, Addison-Wesley, 1992.
3. Selected papers.

References :

1. Bertrand Meyer, Object-Oriented Software Construction, Prentice Hall International series in Computer Science, 1988.
2. Grady Booch, Object Oriented Design with Aplications, Benjamin Cummings, 1991.
3. Brad J. Cox, Object Oriented Programming An Evolutionary Approach, Addison-Wesley.
4. Peter & Wegner (editors) Research Directions in Object-Oriented Programming, MIT Press.
5. Selected papers.

Topics to be covered :

1. Principle of Object-Oriented Software Construction
2. Introduction to C++
3. Classes, methods, messages, instances, inheritance and binding
4. Inheritance and subtyping
5. Polymorphism
6. Inheritance model
7. Theorical models

Programming :

During the course, the students will design and implement several programs in C++.

Apéndice B

Tabla comparativa de libros de texto

	[TUR 88]	[MEY 88]	[ELL 90]	[COX 91]	[BOO 91]	[KHO 90]
MODULARIDAD		1			3	3
OBJETO, ATRIBUTO	1	2	2	8	15	2
ANALISIS		18			6	
DISEÑO		3	1	13	1	
MENSAJE		16	6	9	7	5
ENCAPSULACION	4	14	11	4	4	4
CLASES		5	10	5	13	10
CLASES GENERICAS		7	13		17	
METODO	3	17	5	10	2	6
POLIMORFISMO	6	13	9	15	14	1
HERENCIA	2	11	4	11	11	11
HERENCIA MULTIPLE		12	7	12	12	12
CLASES ABSTRACTAS	5	9	8	14	16	9
TIPO ABSTR. D' DATO	7	4	4	2	5	7
CONSTRUCTORES	9	10	3	6	8	14
DESTRUCTORES	8		12	7	9	15
ASERCIONES		8			20	8
RELACIONES DE USO	10				19	
ENCUBRIMTO. D INF.		15		3	10	13
SISTEMAS		6		1	18	16
LENGUAJES QUE USAN	*O. PASCAL*	*EIFFEL*	*C++*	*SMALLTALK* *ADA* *C++* *OBJ-C*	*SMALLTALK* *O. PASCAL* *C++* *LISP* *ADA*	*C++* *ADA*

*: Da los temas en cuatro lenguajes a la vez.

Nota: Los números indican el orden dado en cada libro.

	[RUM 91]	[LAL 90]	[MEY 92]	[MAR 92]	[WIR 90]	[BUD 91]*
MODULARIDAD	8			10		5
OBJETO, ATRIBUTO	2	6	6	1	6	1
ANALISIS	12		13	15	4	
DISEÑO	14		1	16	1	7
MENSAJE		3	9	5	7	3
ENCAPSULACION	1	1		3	3	8
CLASES	3	7	8	6	9	9
CLASES GENERICAS	16		5			14
METODO	4	2		2	8	2
POLIMORFISMO	5	8	4	9	10	13
HERENCIA	7	9	11	7	11	12
HERENCIA MULTIPLE	10	11	12	8	14	17
CLASES ABSTRACTAS	9	10			12	4
TIPO ABSTR. D' DATO	11	4	2	11	2	6
CONSTRUCTORES				13		10
DESTRUCTORES				14		11
ASERCIONES	17		3			16
RELACIONES DE USO	6		10	12	13	
ENCUBRIMTO. D INF.	15	5	13	4	5	15
SISTEMAS	13		7			
LENGUAJES QUE USAN	*C++ EIFFEL SMALLTALK*	*SMALLTALK*	*EIFFEL*			*O. PASCAL SMALLTALK OBJ-C C++*

*: Da los temas en cuatro lenguajes a la vez.

Nota: Los números indican el orden dado en cada libro.

	[TAY 90]	[COA 91]	Libros que lo usan	% de libros que lo usan	Promedio Lugar	Propuesta Libros	Propuesta Analizada
MODULARIDAD	1		7	50,00%	4,43	2	
OBJETO, ATRIBUTO	3	1	14	100,00%	4,00	1	1
ANALISIS		2	7	50,00%	10,00	14	
DISEÑO		3	10	71,43%	6,00	6	5
MENSAJE	5	7	12	85,71%	6,83	7	6
ENCAPSULACION	8	5	13	92,86%	5,38	3	2
CLASES	6	9	13	92,86%	7,69	8	7
CLASES GENERICAS			6	42,86%	12,00	19	
METODO	4	8	13	92,86%	5,62	5	4
POLIMORFISMO	10		13	92,86%	9,00	10	9
HERENCIA	7	6	14	100,00%	8,57	9	8
HERENCIA MULTIPLE	11	11	13	92,86%	11,46	17	12
CLASES ABSTRACTAS	12	10	12	85,71%	9,83	13	11
TIPO ABSTR. D' DATO	9	4	14	100,00%	5,57	4	3
CONSTRUCTORES		12	9	64,29%	9,44	12	
DESTRUCTORES		13	8	57,14%	11,13	16	
ASERCIONES			6	42,86%	12,00	20	
RELACIONES DE USO			6	42,86%	11,67	18	
ENCUBRIMTO. D INF.	2		11	78,57%	9,09	11	10
SISTEMAS			6	42,86%	10,17	15	
LENGUAJES QUE USAN		*C++ O. PASCAL SMALLTALK OBJ-C EFFEL ADA*					

*: Da los temas en cuatro lenguajes a la vez.

Nota: Los números indican el orden dado en cada libro.

Apéndice C

Encuesta

Apéndice C-I

Formato de la encuesta

INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS ESTADO DE MEXICO

ENCUESTA:
PLANEACION CURRICULAR DE POO

MARQUE CON UNA CRUZ O RESPONDA SOBRE LA LINEA POR FAVOR:

¿QUE SEMESTRE CURSAS?: _____
¿DE QUE CARRERA?: _____

¿HA LLEVADO ALGUN CURSO DE POO (Programación Orientada a Objetos)?
SI NO

SI NO LO HA LLEVADO, ¿LE GUSTARIA LLEVAR EN SU CARRERA ALGUN CURSO DE POO?
SI NO

SI NO HA LLEVADO ALGUN CURSO DE POO, YA NO CONTINUE Y ENTREGUE SU ENCUESTA.
GRACIAS POR TODO.

¿EN QUE SEMESTRE LLEVO POO? _____

¿CUALES LENGUAJES VIO?:
EIFFEL C++ SMALLTALK OTRO:_____

¿HUBIERA PREFERIDO VER ALGUN OTRO LENGUAJE APARTE DEL QUE VIO?
SI NO

¿CUAL(ES)?: _____

¿CONSIDERA QUE APRENDIO A ESTRUCTURAR UN PROBLEMA CON CLASES Y OBJETOS?
SI NO

¿Y A PROGRAMAR EN OBJETOS?
SI NO

¿POR QUE?
_____
_____
_____

¿HUBIERA PREFERIDO MAS CLASES DE TEORIA QUE DE PRACTICA?
SI NO

¿POR QUE?
_____
_____
_____

¿PODRIA MEJORAR EN ALGO AL PLAN DE ESTUDIOS DE LA MATERIA QUE LLEVO?
SI NO

SI SU RESPUESTA A LA PREGUNTA ANTERIOR FUE "SI", ¿QUE MEJORARIA?

_____
_____
_____

EN QUE ORDEN PONDRIA LOS SIGUIENTES TEMAS PARA DARLOS EN CLASE:
(COLOQUE NUMEROS PARA INDICAR EL ORDEN Y UNA CRUZ PARA INDICAR QUE NO SE DEBERIA DAR)

_____ DEFINICION DE OBJETO, CLASE, METODO, MENSAJE.
_____ DEFINICION DE MODULARIDAD, HERENCIA, HERENCIA MULTIPLE.
_____ ANALISIS Y DISEÑO DE SISTEMAS ORIENTADOS A OBJETOS.
_____ LENGUAJE EIFFEL.
_____ LENGUAJE SMALLTALK.
_____ LENGUAJE C++.
_____ INTERACCION LENGUAJE-AMBIENTE GRAFICO.
_____ OTRO TEMA:_____

¿ALGUN COMENTARIO EXTRA SOBRE LA MATERIA?

_____
_____
_____

¡GRACIAS POR SU TIEMPO Y AYUDA PARA SER CADA VEZ MEJORES!!!

Apéndice C-II

Tabla de resultados de la encuesta

CARRIERA:	Total	SEMESTRE ACTUAL:	Cant.	HA CURSADO POO?		Cant.
LSCA :	31	SEXTO	10	SI	semestre: 6	5
				NO	Gustaria	5
					No gustaria	
		SEPTIMO	13	SI	semestre: 6	13
				NO	Gustaria	
					No gustaria	
		OCTAVO	2	SI	semestre: 7	2
				NO	Gustaria	
					No gustaria	
		NOVENO	6	SI	semestre: 7	6
				NO	Gustaria	
					No gustaria	
ISC :	5	SEPTIMO	1	SI	semestre: 6	1
				NO	Gustaria	
					No gustaria	
		OCTAVO	3	SI	semestre: 5	3
				NO	Gustaria	
					No gustaria	
		NOVENO		SI	semestre: 0	
				NO	Gustaria	
					No gustaria	
		GRADUADO	1	SI	semestres 4 y 9	1
				NO	Gustaria	
					No gustaria	
TOTAL :	36					

LSCA : Licenciado en Sistemas de Computación Administrativo
ISC : Ingeniero en Sistemas de Computación

Nombre del lenguaje:	EIFFEL	C++	SMALLTALK	ACTOFI	LISP	PASCAL OO
Vieron el lenguaje	26	25	20	1	1	0
% de los que sí vieron	83,87%	80,65%	64,52%	3,23%	3,23%	0,00%
No lo vieron y quieren verlo	1	4	5	3	0	2
	20,00%	66,67%	45,45%	10,00%	0,00%	6,45%
% q´lo quiere ver (ya vieron 3)	0,00%	0,00%	0,00%	0,00%	0,00%	5,56%
% q´lo quiere ver (del resto)	7,69%	30,77%	38,46%	23,08%	0,00%	7,69%

	SI	No
Lenguajes vistos	31	5
% del Total (36 personas)	86,11%	13,89%

Vieron 3 lenguajes en una materia:	18	13

Hubiera preferido otro:	12	19
de los que vieron 1:	10	3
	83,33%	15,79%
de los que vieron 3:	2	16
	16,67%	84,21%

Aprendió a estructurar pro-	24	7
blemas con clases u objetos	77,42%	22,58%
Aprendió a programar	18	13
en objetos	58,06%	41,94%

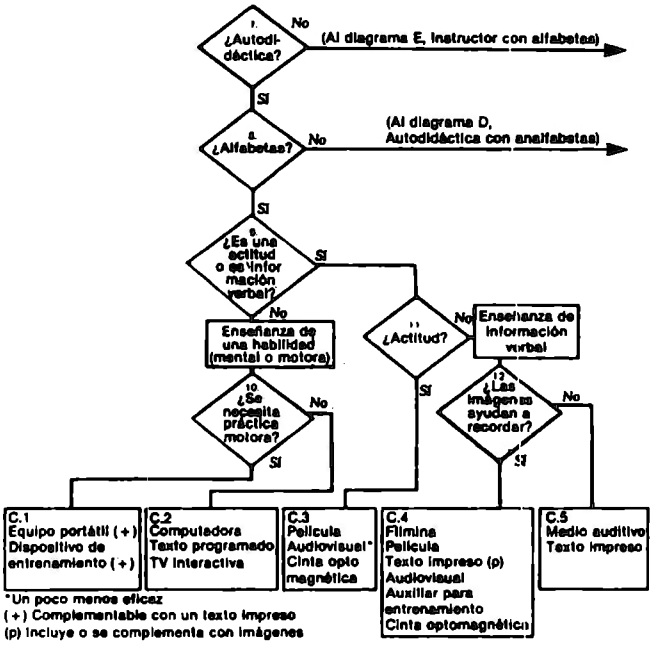
Más Teoría que Práctica?	10	21
	32,26%	67,74%

TEMA:	1	2	3	4	5	6	7	8	% de Petición:	Pro-puesta
Concepto de Objeto, Clase Método, Mensaje	24 77,4%	6 19,4%	1 3,0%						100,0%	1
Concepto de Modu-laridad, Herencia y Herencia Múltiple	2 6,5%	19 61,3%	7 21,2%	2 6,5%	1 2,8%				100,0%	2
Análisis y Diseño de Sistemas O-O	5 16,1%	6 19,4%	19 57,6%		1 2,8%				100,0%	3
Lenguaje Eiffel			1 3,0%	12 38,7%	9 25,0%	6 23,1%	2 9,1%		96,8%	4
Lenguaje Smalltalk			3 9,1%	4 12,9%	11 30,6%	8 30,8%	2 9,1%		90,3%	6
Lenguaje C++			1 3,0%	10 32,3%	9 25,0%	6 23,1%	2 9,1%		90,3%	5
Interacción Lenguaje-Ambiente Gráfico			1 3,0%	2 6,5%	5 13,9%	6 23,1%	16 72,7%	1 33,3%	100,0%	7
Otro Tema: Aplicaciones Prácticas								1 33,3%	3,2%	8
Otro Tema: Interacción entre Lenguajes								1 33,3%	3,2%	8
Otro Tema: Metodología de Sistemas				1 3,2%					3,2%	8

Propuestas:	daria en dos materias	más diseño	más programas	ir más lento.	menos lenguajes	más teoría	más ejs. resueltos	más bibliografía	más análisis
Cantidad:	5	3	9	2	7	7	1	2	2
	16,1%	9,7%	29,0%	6,5%	22,6%	22,6%	3,2%	6,5%	6,5%
Alumnos con:									
3 lenguajes	0	2	2	2	7	5	0	1	1
% de Cantidad	0,0%	66,7%	22,2%	100,0%	100,0%	71,4%	0,0%	50,0%	50,0%
1 o 2 lengs.	5	1	7	0	0	2	1	1	1
% de Cantidad	100,0%	33,3%	77,8%	0,0%	0,0%	28,6%	100,0%	50,0%	50,0%

Apéndice D

Diagrama para la selección de medios que ayuden al aprendizaje


7. ¿Autodidáctica?
No
(Al diagrama E, Instructor con alfabetas)
Si
8. ¿Alfabetas?
No
(Al diagrama D, Autodidáctica con analfabetas)
Si
9. ¿Es una actitud o es información verbal?
Si
No
Enseñanza de una habilidad (mental o motora)
10. ¿Se necesita práctica motora?
No
Si
11. ¿Actitud?
No
Si
Enseñanza de información verbal
12. ¿Las imágenes ayudan a recordar?
No
Si
C.1 Equipo portátil (+) Dispositivo de entrenamiento (+)
C.2 Computadora Texto programado TV interactiva
C.3 Película Audiovisual* Cinta opto magnética
C.4 Filmina Película Texto impreso (p) Audiovisual Auxiliar para entrenamiento Cinta optomagnética
C.5 Medio auditivo Texto impreso
*Un poco menos eficaz
(+) Complementable con un texto impreso
(p) Incluye o se complementa con imágenes

Apéndice E

Recursos

Apéndice E-I

Humanos

Para la elaboración de esta tesis se contó con la colaboración de tres asesores, cada uno con especialidad en una de las áreas que involucraba este trabajo, a saber, el ingeniero Humberto Cárdenas Anaya con maestría en Sistemas de Información y director de las carreras de Licenciado en Sistemas Computacionales Administrativos e Ingeniero de Sistemas de Información, como soporte en el paradigma de objetos y cuyos alumnos se ven directamente afectados con los resultados aquí obtenidos.

El ingeniero Joaquín Méndez Almaguer con maestría en Educación y director del Centro de Calidad del Campus para asegurarnos de hacer un trabajo con calidad y que cumpla los objetivos propuestos. El tercer asesor es el psicólogo Rodrigo Polanco Bueno con maestría en Ciencias y estudiando actualmente el doctorado en Educación, especialista del departamento de Desarrollo Académico en educación, para garantizar la funcionalidad de esta tesis.

Como soporte se contó con el estudiante Julio César Munive Ramírez que laboró como parte de su servicio social por un tiempo de trescientas (300) horas; su trabajo consistió en la aplicación y procesamiento de las encuestas, obtención de la información de otras universidades de la Ciudad de México acerca de su forma de impartir el curso de programación orientada a objetos si lo hacían, otros trabajos de recopilación de información y soporte en la elaboración del material del tutorial.

Otros que nos ofrecieron su ayuda para la obtención de un tutorial de alta calidad fueron: María Antonieta Reynoso, Marciano Alberto Moreno, el ingeniero Juan Manuel Montoya, el diseñador gráfico Francisco Javier Palmerín y Judith Cortés, los dos primeros alumnos del Instituto y los otros tres miembros del personal del Campus.

Un agradecimiento muy especial a mis alumnos de la materia de Programación Orientada a Objetos, que fueron partícipes de este proceso de mejora.

Apéndice E-II

Técnicos

Las requisiciones de equipo realmente fueron pocas, ya que el trabajo consistió en gran medida de recopilar información, estudiarla y analizarla. El mayor esfuerzo utilizando la computadora, se aplicó casi todo en el tutorial. El equipo utilizado y su propósito se describe a continuación.

a) Computadora IBM, sobre la cual se procesó la información de las encuestas, las comparaciones y se escribió el documento. Con la siguiente configuración:

Modelo PS/257

Procesador Intel 80386

4 MB en RAM

Monitor VGA a Color

Teclado extendido

Mouse

160 MB en disco duro

b) Computadora Macintosh, sobre la cual se desarrolló el prototipo en multimedia del tutorial. Con la siguiente configuración:

Modelo Quadra 900

Procesador Motorola 68040

16 MB en RAM

Monitor de 21'

Teclado extendido

Mouse

500 MB en disco optico externo

130 MB en disco duro

c) Cámara de video utilizada para la filmación de algunos videos que aparecen en el tutorial.

d) Videograbadora con formato VHS que sirvió como puente entre la cámara y la computadora para digitalizar el video.

El software fue variado, tratando de aprovechar el mejor para cada una de las cosas que se necesitaban hacer video, sonido e imagenes, sobre los que se estuvo trabajando fue el siguiente:

i) Photoshop versión 2.0.1 para Macintosh de Adobe Systems Incorporated
ii) Premier TM para Macintosh de Adobe Systems Incorporated
iii) Excel versión 3.0 para Windows de Microsoft
iv) Infini-D versión 1.1 para Macintosh de Specular International
v) Director versión 3.1 para Macintosh de MacroMind
vi) Painter versión 2.0 para Macintosh de Fractal Designing Corporation
vii) Sound Edit TM Pro versión 1.0 para Macintosh de MacroMind
viii) Word versión 2.0 para Windows de Microsoft

A28