Universidad Tecnológica de la Mixteca

TESIS

Herramienta Distribuida para Fortalecer el Proceso de Aprendizaje de las Matemáticas de Sexto Año de Primaria mediante Tecnología CORBA

Para obtener el título de:
INGENIERO EN COMPUTACIÓN

Presenta:
FRANCISCO JAVIER HERNÁNDEZ REYES

Director de tesis:
M.C. EVERTH HAYDEÉ ROCHA TREJO

Huajuapan de León, Oax. Octubre de 2005.

A ti mamá

Por representar la unión y amor familiar que me han permitido alcanzar esta meta, Lucy este logro también es tuyo.

A ti papá

Por que tu lucha constante y fuerza de voluntad han sido el mejor ejemplo de superación que he recibido, Napo gracias a ti alcancé este sueño.

A lizita

Por que tus consejos y las hermosas charlas que compartimos siempre me animan a no caer, hermana, gracias por ser mi guía.

A juvito

Con todo mi cariño te dedico este logro esperando que te motive a alcanzar los tuyos, hermano, gracias por alegrar mi vida.

A mis abuelos, tías y primos

Por que me acompañan y velan por mí haciéndome sentir un miembro más de sus familias, mil gracias.

Los amo a todos.

Agradecimientos

En primera instancia agradezco a mi casa de estudios la UTM, por haber cimentado las bases de mi formación profesional.

Agradezco enormemente a mi asesora y amiga la M.C. Everth Haydeé Rocha Trejo, por el apoyo y los valiosos consejos que me brindó durante el desarrollo de este proyecto.
Everth, sin ti no hubiera sido posible.

Un especial agradecimiento a mis sinodales M.C. Mónica Edith García García, M.C. David Martínez Torres, y en particular al M.C. Gabriel Gerónimo Castillo, por haber enriquecido este trabajo.
Gabriel, gracias por tu apoyo de principio a fin.

Le agradezco al M.C. Mario Alberto Moreno Rocha, por haber colaborado brindándome parte de su experiencia en el diseño de interfaces, además de facilitarme el uso de las instalaciones del LIDIS y del UsaLab.

Gracias al D.G. Jorge Vázquez Sánchez, por su valiosa colaboración en el diseño gráfico de la interfaz de la herramienta que en este proyecto se ha gestado.

Extemo mi gratitud a las escuelas primarias, y a los docentes que me apoyaron durante la investigación de campo que requería esta tesis.

Mil gracias a la profesora de educación primaria, Lic. Ángeles Trujillo Reyes, así como a los niños de primaria, y de secundaria por su colaboración en la fase de pruebas de esta herramienta didáctica.

A mis compañeros y amigos les agradezco el haber compartido conmigo experiencias que han alimentado mi vida, y por hacer los momentos difíciles más llevaderos.

Gracias Chuchin, por haberme permitido alcanzar esta meta.

Contenido

Lista de figuras

Lista de tablas

Prólogo

Los problemas educativos que enfrenta México, lo han comprometido a construir una educación de calidad, sustentada en la investigación, en la aplicación de metodologías didácticas, herramientas, y prácticas educativas innovadoras que fortalecen los procesos de enseñanza - aprendizaje.
La expansión acelerada de las nuevas tecnologías de información y comunicación, representan una oportunidad para el desarrollo educativo. El uso de nuevas prácticas educativas basadas en tecnologías de cómputo, es posible gracias al desarrollo de software.

La presente tesis, tiene como objetivo fortalecer el proceso de aprendizaje de algunos subtemas de las matemáticas del plan de estudios del sexto grado de primaria, propuesto por la Secretaría de Educación Pública, mediante la creación de una herramienta basada en las metodologías de enseñanza lúdica y cibernética, y en el uso de tecnología distribuida, el uso de la herramienta implantará una práctica educativa innovadora. La herramienta utiliza como plataforma de desarrollo el sistema operativo Linux en su distribución Fedora Core 2, y MICO que es una implementación libre basada en CORBA.

Este documento está organizado en siete capítulos, el contenido de cada uno de ellos, se explica a continuación:

Capítulo 1. La educación básica en México. Brinda un panorama general de la educación básica en México, también presenta la investigación realizada que identificó la existencia de problemas educativos en la enseñanza y el aprendizaje de las matemáticas. Propone abordar el problema con un enfoque metodológico distinto, en el que intervenga una herramienta que emplee tecnología informática.

Capítulo 2. Análisis y obtención de requisitos. Expone los resultados de la investigación que permitió, conocer la forma en que la SEP sugiere abordar los subtemas de las matemáticas a los que va dirigido la herramienta. Presenta el análisis de distintas herramientas de software didácticas en el que se concluye que, ninguna

se apega a lo que estipula la SEP. En base a esta información, se analizan y obtienen los requisitos de la herramienta.

Capítulo 3. Arquitectura de distribución del sistema. Trata la tecnología empleada para la implantación de la herramienta, y presenta detalladamente la arquitectura de distribución del sistema.

Capítulo 4. Modelado de la arquitectura del sistema. Muestra el modelado de la arquitectura del sistema, comenzando con el modelado de la interfaz de los objetos distribuidos que provee la herramienta, hasta los diagramas de clase y de secuencia de la misma.

Capítulo 5. Implementación del sistema. Documenta los aspectos más relevantes de la implementación del sistema, desde la perspectiva del cliente, y del servidor.

Capítulo 6. Modelado de la interfaz del usuario. Presenta el proceso de modelado realizado a la interfaz de la herramienta, y habla de las metodologías de usabilidad empleadas para mapear el modelado a elementos externos.

Capítulo 7. Pruebas, mejoras y resultados. Habla de las pruebas de usabilidad realizadas al sistema, y muestra las mejoras hechas conforme a las sugerencias generadas en estas pruebas. Finalmente presenta distintas pantallas de la herramienta, las cuales, se obtuvieron al término de este trabajo.

Capítulo 1

La educación básica en México

En toda sociedad moderna, la educación es considerada un factor de gran importancia, ya que es fundamental para su óptimo desarrollo. Sin embargo, en México la educación afronta tres grandes dificultades que se expresan como *educación para todos*, *educación de calidad* y *educación de vanguardia* [25].

En la educación primaria, las dificultades educativas se reflejan en el bajo rendimiento escolar, que se concibe en el escaso nivel de conocimientos de un alumno, en una baja eficiencia terminal, en la deserción escolar y en el alto índice de reprobación. Estos aspectos, muestran la carencia de calidad educativa que obstaculiza el desarrollo de nuestro país. Para enfrentar las dificultades, se han realizado reformas a los planes y programas de estudio de la educación básica mexicana. Las reformas comprenden el fortalecer las capacidades de lectura, *el manejo de las matemáticas* y el conocimiento básico de las ciencias, proceso que toma lugar principalmente en las escuelas primarias. Desafortunadamente, gran parte de las causas que originan el bajo rendimiento escolar se encuentran en los procesos de enseñanza que tienen lugar en el aula [24].

1.1 Metodologías de enseñanza, herramientas didácticas y prácticas educativas innovadoras

Con el afán de mejorar los procesos de enseñanza, distintos organismos se han dado a la tarea de investigar y aplicar metodologías didácticas, prácticas educativas innovadoras, así como el uso adecuado de herramientas que fortalezcan y faciliten tanto la enseñanza como el aprendizaje de diversas materias. Dentro de estos organismos se encuentran: La Sociedad Europea en Educación Matemática (*European Society in Mathematics Education*), el grupo de trabajo sobre Representaciones y Visualizaciones Matemáticas (*Working Group on Representations and Mathematics Visualization*) de la Universidad Estatal de Carolina del Norte en Estados Unidos, el Ministerio de Educación de Chile y la Secretaría de Educación Pública (SEP) de México.

La presente investigación ha hallado las siguientes metodologías de enseñanza y herramientas [7] [12] [19] [20]: *Método Autogénico*, *Método Interpersonal*, *Método Intrapersonal*, *Método Lúdico*, *Método Platónico*, *Cibernética*, *Metáforas e Imágenes*, y *Observación*. Éstas metodologías se sugieren como las principales para aplicar en la enseñanza.

Adicionalmente, las prácticas educativas que se han implantado en México, incorporan una o más metodologías y/o herramientas didácticas, se relacionan con la gestión escolar, las nuevas tecnologías de la información, los métodos y la planeación educativa [26].

Algunos ejemplos de las prácticas educativas que se llevan a cabo en nuestro país son: *Estrategia pedagógica para incrementar la habilidad de comprensión lectora en los alumnos de educación primaria* (Guerrero), *Ludoteca interactiva de matemáticas para la educación secundaria* (Hidalgo), e *Integración tecnológica: prensa, radio, televisión e informática al servicio de la educación básica* (Sonora).

El uso apropiado de las metodologías, herramientas, y prácticas educativas, fomenta una educación de calidad, debido a que permiten abordar la problemática educacional con un enfoque novedoso y atractivo, apoyando la labor de enseñanza de los docentes, y facilitando el aprendizaje de los alumnos en diversos ejes[1] del conocimiento [26].

El presente trabajo, tiene como principal objetivo fortalecer el proceso de aprendizaje de algunos subtemas de las matemáticas basado en el plan de estudios del sexto grado de primaria propuesto por la SEP, mediante la creación e implantación de una herramienta didáctica que incorpore tecnología distribuida bajo el estándar CORBA (Arquitectura Intermediaria para la Solicitud de Objetos Compartidos - *Common Object Request Broker Architecture*).

[1] Un eje es un medio que utiliza la SEP para organizar la didáctica, cada eje del plan y programa de estudios, se divide en contenidos que se integran por subtemas.

Para el logro de este objetivo, es necesario el cumplimiento de los siguientes objetivos parciales:

1. Detectar el eje y contenido de las matemáticas de sexto grado de primaria, en que se presenta mayor dificultad de aprendizaje y/o enseñanza.
2. Reconocer el conjunto de aplicaciones didácticas existentes para el apoyo al contenido detectado en el objetivo anterior.
3. Definir la especificación de la propuesta en base a la información recolectada en los objetivos previos.
4. Modelar y desarrollar la herramienta que apoye el aprendizaje y/o enseñanza de algunos subtemas asociados al contenido detectado en el primer objetivo específico.
5. Modelar y desarrollar la interfaz de la herramienta en base a las necesidades de sus usuarios.
6. Realizar pruebas de usabilidad y del sistema.

1.2 El problema educativo en la enseñanza de las matemáticas

Para cumplir con las metas propuestas, el primer paso fue recabar información que diera indicio del eje de las matemáticas, en el cual, la herramienta pudiera brindar un beneficio tangible. Por lo tanto, se realizó un estudio que consistió en aplicar encuestas en diversas instituciones educativas de nivel básico, para detectar el eje con mayor complejidad de enseñanza y/o aprendizaje.

A continuación se presentan los seis ejes que integran los contenidos matemáticos del sexto grado de educación básica en México [23]:

1. *Los números, sus relaciones y sus operaciones*. Los contenidos de este eje, permiten al alumno comprender el significado de los números y las relaciones que pueden establecerse entre ellos, para utilizarlos en la resolución de problemas.
2. *Medición.* Ayuda al alumno a construir los conceptos ligados con la medición, a través de acciones directas sobre objetos reales, mediante la reflexión de esas acciones y la comunicación de los resultados.
3. Geometría. Presenta contenidos y situaciones que favorecen la ubicación del alumno en relación con su entorno. Asimismo propone actividades de manipulación, observación, dibujo y análisis de formas diversas.

4. Procesos de cambio. En él, se abordan fenómenos de variación proporcional y no proporcional. Se conforma por la lectura, elaboración, análisis de tablas y gráficas donde se registran y analizan procesos de variación. Además, cubre las nociones de razón y proporción.
5. Tratamiento de la información. Ofrece situaciones que promueven el análisis y selección de información planteada a través de textos, imágenes u otros medios. Esto da habilidad al alumno para comprender y resolver problemas con mayor facilidad.
6. Predicción y azar. Pretende que los alumnos exploren situaciones donde interviene el azar, para desarrollar la noción de lo probable e improbable.

Paralelamente, el estudio ayudó a conocer e identificar las metodologías de enseñanza empleadas por los profesores para impartir las matemáticas a nivel básico.

El universo poblacional del estudio, se conformó por los treinta y seis profesores que imparten el sexto grado de primaria en las veinticuatro escuelas de la ciudad de Huajuapan de León, Oaxaca, México. Las encuestas se aplicaron a veintitrés profesores de quince escuelas, con lo que se obtuvo una confianza del 95% y un error máximo de estimación cercano al 10%.

El tamaño de la muestra poblacional, que garantiza una certidumbre confiable en la estimación, se obtuvo con la siguiente ecuación[2]:

$$n = \frac{s^2 N p q}{e^2 (N-1) + s^2 p q}$$

Donde:

N =Universo poblacional
n = Tamaño de la muestra
s = Nivel de confianza
e = Error máximo de estimación
p = Probabilidad a favor
q = Probabilidad en contra

Las quince escuelas que participaron en el análisis se listan a continuación:

1. 21 De Marzo
2. 23 De Julio
3. Anton Semyonovich Makarenko

[2] Laura Fischer. Introducción a la investigación de mercados, Cálculo del tamaño de la muestra para poblaciones finitas.

4. Año de Juárez
5. Colegio Teresa Martín
6. Coronel Valerio Trujano
7. Emiliano Zapata
8. General Antonio de León
9. Ignacio Manuel Altamirano
10. Maestro Ju sto Sierra
11. Manuel González Gatica
12. Mundo Mágico
13. Presidente Lázaro Cárdenas
14. Trabajadores del Campo
15. Valentín Gómez Farias

Cabe señalar que estas escuelas se encuentran incorporadas a la Secretaría de Educación del estado de Oaxaca.

Dentro de los resultados arrojados por el estudio, se puede destacar información de interés. Referente al aprendizaje de las matemáticas, el estudio identificó como el eje de mayor complejidad al titulado “Los números, sus relaciones y sus operaciones”. En la figura 1.1, se observa que el 48% de los profesores considera que este eje es el que causa mayor problema de aprendizaje a los alumnos.

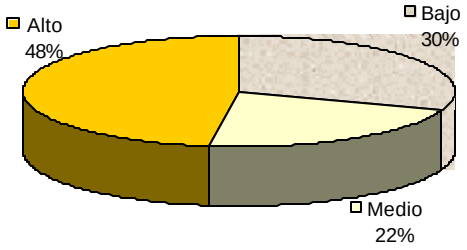

Figura 1.1. Nivel de dificultad de aprendizaje del eje los números, sus relaciones y sus operaciones.

De igual manera, se determinó que de los tres contenidos que integran al eje identificado, el problema principal de aprendizaje se centra en el contenido llamado “Los número fraccionarios”, tal como se muestra en la figura 1.2. Endonde se observa que el 87% de los profesores opina que los alumnos presentan mayor dificultad de aprendizaje en este contenido.

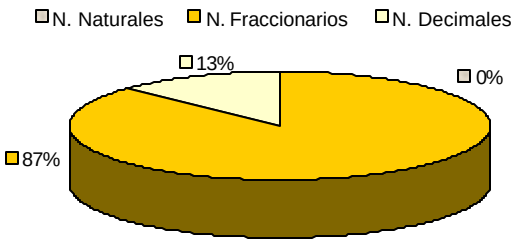

Figura 1.2. Dificultad de aprendizaje de los contenidos del eje los números, sus relaciones y sus operaciones.

En relación con la enseñanza de las matemáticas, los profesores no consideran tener gran dificultad para la docencia del eje "Los números sus relaciones y sus operaciones". La figura 1.3 muestra que solamente el 39% de ellos consideran a este eje de difícil enseñanza.

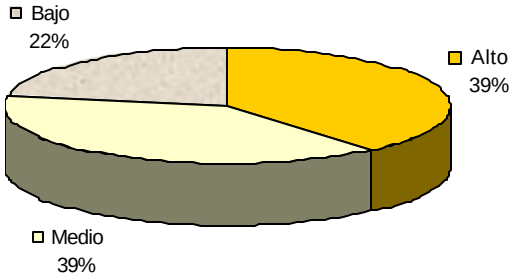

Figura 1.3. Dificultad de enseñanza del eje los números, sus relaciones y sus operaciones.

Con respecto a las metodologías de enseñanza empleadas para las matemáticas, el estudio dio a conocer técnicas de docencia diferentes a las descritas previamente. Estos métodos de enseñanza son: *Método Constructivista*, *Método Crítico*, *Método Ecléctico*, *Método Funcional*, *Método Inductivo – Deductivo*, y *Método Operante*.

A pesar de la existencia de una gran diversidad de metodologías de enseñanza, son pocos los profesores que las conocen y hacen uso de ellas. El estudio hizo notar que la mayoría de los docentes se apoya principalmente del método inductivo - deductivo para la enseñanza de las matemáticas, tal como se muestra en la figura 1.4, mientras que únicamente el 30% de ellos conoce y hace uso de métodos distintos.

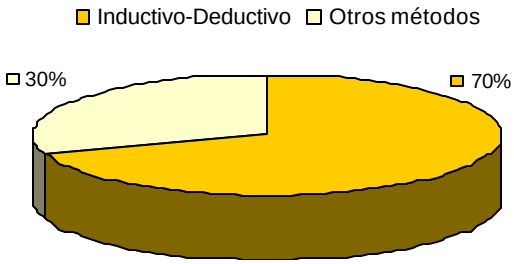

Figura 1.4. Metodologías de enseñanza para las matemáticas empleadas en las escuelas primarias.

La información rescatada del análisis es contradictoria, y obliga a plantear el siguiente cuestionamiento, ¿cómo es posible que un eje que es fácil de enseñar por los docentes, sea a la vez de difícil aprendizaje para los alumnos?, todo parece indicar que el problema radica en que la metodología de enseñanza empleada no es la más adecuada.

Para hacer frente al problema expuesto, es recomendable emplear una metodología de enseñanza de las fracciones distinta.

1.3 La tecnología como soporte educativo

La presente tesis propone abordar el problema expuesto, mediante una práctica educativa innovadora que proporcione soporte a las necesidades de aprendizaje del eje "Los números, sus relaciones y sus operaciones", específicamente al contenido "Los números fraccionarios", el cuál está integrado por los siguientes subtemas [23]:

- Ubicación de fracciones en la recta numérica.
- Equivalencia y orden entre fracciones.
- Planteamiento y resolución de problemas de suma y resta de fracciones mixtas.
- Conversión de fracciones mixtas a impropias y viceversa.
- Simplificación de fracciones.
- Planteamiento y resolución de problemas de suma y resta de fracciones con denominadores distintos mediante el cálculo del denominador común.

Debido al amplio margen de estudio de este contenido, la propuesta planteada, únicamente se enfoca a dos de los seis subtemas que lo componen. Dichos subtemas son "Equivalencia y orden entre fracciones" y "Simplificación de fracciones".

Para el desarrollo de la práctica educativa, se ha propuesto el uso de metodologías de enseñanza lúdicas, cibernéticas, así como el uso de imágenes. Paralelamente se hace uso de tecnología distribuida para la implantación de la herramienta.

En este punto, es necesario recalcar que las tecnologías de información y comunicación juegan un papel estratégico de gran apoyo a la instrucción básica, pues la informática ofrece recursos para la presentación de escenarios que facilitan y mejoran el aprendizaje [22]. Dentro de las ventajas que tiene el uso de tecnología en la educación, se encuentran [16] [9]:

Mejor aprendizaje. El aprendizaje se mejora debido al impacto visual que proveen las tecnologías computacionales, ya que generalmente incorporan la manipulación de imágenes y sonido, además, permiten el uso de técnicas de enseñanza innovadoras a los profesores.

Motivación y compromiso. La motivación y el compromiso de los alumnos con su aprendizaje se ven incrementados.

Aprendizaje fuera de clase. El acceso en línea a los materiales del curso, otorga a los alumnos mayor control sobre su aprendizaje, y posibilita el aprendizaje fuera de las aulas y horarios de clase.

Auto-evaluación. Los alumnos se concentran en discusiones, juegos y técnicas mentales. Por lo tanto, están más pendientes de su propio aprendizaje.

Es debido a estas ventajas, que se ha decidido hacer uso de la tecnología para la implantación de la herramienta didáctica.

Capítulo 2

Definición de las actividades de la herramienta

El propósito de este capítulo es satisfacer el segundo y tercer objetivo específico, para reconocer las aplicaciones didácticas existentes que apoyan el aprendizaje de las fracciones, y definir la especificación de la propuesta.

2.1 Análisis de aplicaciones didácticas que apoyan el aprendizaje de las fracciones

Para cumplir con el segundo objetivo específico, se plantearon metas que tras su logro permitieran reconocer si las aplicaciones didácticas existentes, brindan apoyo al aprendizaje de las fracciones conforme a los planes y programas de estudio de la educación básica mexicana. Las metas que se plantearon son:

1. Recabar los objetivos que la SEP propone cumplir en los subtemas "Equivalencia y orden entre fracciones" y "Simplificación de fracciones".
2. Determinar la forma de enseñanza sugerida por la SEP para introducir los subtemas a los alumnos.

2.1.1 Análisis de los subtemas y de su forma de enseñanza

Para llevar a buen término las metas, fue necesario el estudio de diversas fuentes bibliográficas distribuidas por la SEP [12] [13] [14]. Además, se revisaron otros libros que permitieron complementar y comprender de mejor manera la información recabada [8].

Los objetivos particulares de cada subtema, así como la forma en que se sugiere deben presentarse al alumno, se comentan a continuación:

Equivalencia de fracciones. El objetivo de este subtema consiste en que los alumnos comprendan que obtener fracciones equivalentes es encontrar expresiones simbólicas que representan la misma cantidad [12]. La SEP recomienda iniciar su

estudio con actividades que involucren la partición de superficies, para culminar con algún método numérico.

Orden entre fracciones. En este subtema, se le brindan conocimientos al alumno para que esté en condiciones de determinar si un número racional es menor, mayor o igual a otro. En [8], se sugiere iniciar su estudio partiendo de la representación de los números racionales en la recta numérica, para posteriormente introducir el método numérico.

Simplificación de fracciones. Aquí, el objetivo es que el alumno comprenda que la simplificación, significa obtener otra fracción equivalente cuyo numerador y denominador sean menores que los de la fracción original [13]. La simplificación, es un caso particular de la equivalencia, por lo tanto, puede abordarse de forma similar.

En conclusión, los resultados derivados de la revisión bibliográfica, indican que la SEP hace uso de tres formas distintas para presentar estos subtemas a los alumnos, estas formas son:

a. Presentación de subtemas mediante la representación gráfica de información.
b. Presentación de subtemas mediante información textual auxiliada de breves representaciones gráficas.
c. Presentación de subtemas únicamente con información textual.

La SEP recomienda que la información proporcionada en los subtemas, se vincule a contextos cotidianos que faciliten el aprendizaje e incrementen el interés por aprender.

2.1.2 Tecnología aplicada a la educación

Tomando como referencia la información recaudada y los resultados obtenidos, se analizaron recursos de software relacionados con el aprendizaje de las fracciones en los siguientes aspectos: forma de enseñanza aplicada, subtemas abordados, forma de distribución, tipo de software, y plataforma de uso. La mayoría de los recursos estudiados se describe en el apéndice A.

Al finalizar el estudio, se determinó que ninguno de los recursos, obedece completamente las sugerencias estipuladas por la SEP, ya que sus contenidos están

enfocados a alumnos con un nivel particular de conocimiento, además de limitarse a brindar y evaluar información abstracta.

De los recursos analizados, se concluye que aquellos que explotan en mayor medida representaciones gráficas de información, están enfocadas a alumnos cuyo aprendizaje de las fracciones está en sus inicios. Los recursos que hacen un menor uso de representaciones gráficas, e incrementan las tareas mentales del usuario, están diseñados para individuos que gozan de un conocimiento medio de las fracciones. Por último, bs que exigen del usuario cálculos mentales complejos, van dirigidos a alumnos que tienen un dominio amplio del tema.

Las características de los recursos, los hacen poco útiles a usuarios con conocimientos diferentes, por si fuera poco, no fomentan que el alumno aplique su conocimiento a contextos reales. Lo anterior representa un grave problema, pues las facilidades computacionales no son útiles si los estudiantes no saben cuando aplicar los procesos matemáticos que están aprendiendo.

Por lo tanto, es importante que la propuesta ofrezca actividades didácticas enfocadas a usuarios con distinto nivel de conocimiento. Estas actividades deben transportar al alumno a entornos reales que le permitan aplicar su conocimiento matemático en tareas concretas y cotidianas.

2.2 Definición de las actividades de la herramienta didáctica

Finalizada la revisión de los recursos de software, se dio seguimiento al tercer objetivo específico, el cual consiste en definir la especificación de la propuesta de tesis. El primer paso en ésta tarea fue identificar y decidir un escenario sobre el que se desarrollarían las distintas actividades que se planeaban integrar a la herramienta.
Se busca que al utilizar el alumno la herramienta, se genere en él un modelo mental entendible, agradable y familiar. Por tal motivo, el escenario debe cumplir con estos requisitos, además, que permita la aplicación de conocimiento matemático sobre fracciones. Tomando en cuenta las consideraciones antes mencionadas, se propone que las actividades del sistema hagan sentir al alumno inmerso en un mercado mexicano.

La elección de un escenario de este tipo, da libertad al alumno para que se desplace de un puesto a otro, es decir, elija la actividad que desea practicar en base a su necesidad de aprendizaje, pudiendo trabajar únicamente con la que para él represente un reto. Además, al brindar un entorno familiar y concreto, se fomenta que el alumno vincule su conocimiento matemático en entornos del mundo real.

Internamente el sistema debe mantener clasificadas las actividades en tres grupos en base a su contenido. Estos grupos son: “La equivalencia de fracciones”, “Orden de fracciones” y “Simplificación de fracciones”. Dentro de los cuales existen actividades enfocadas a usuarios con distinto nivel de conocimiento .

2.2.1 Actividades de apoyo para el aprendizaje del grupo equivalencia de fracciones

Considerando que la equivalencia de fracciones, de acuerdo al plan y programas de estudio de la SEP, es un tema introducido al alumno desde el cuarto grado de enseñanza básica, y es empleado en diversos contextos en el quinto grado, no es necesario que este grupo proporcione actividades que introduzcan al alumno el significado de la equivalencia de fracciones. Por lo tanto, únicamente incluye actividades que fortalecen las bases de conocimiento del alumno, y fomentan el cálculo mental. Enseguida, se describen las actividades que integra este grupo:

- Actividad intermedia de la equivalencia de fracciones. Permite que el alumno aplique el concepto de equivalencia de fracciones, la actividad se apoya de representaciones gráficas y textuales de información.
- Actividad avanzada de la equivalencia de fracciones. Ésta actividad textual, introduce al alumno el método numérico para la obtención de fracciones equivalentes. Utiliza un mínimo de representaciones gráficas, el alumno debe hacer mayor uso de cálculos mentales.

2.2.2 Actividades de apoyo para el aprendizaje del grupo orden de fracciones

El orden de fracciones, es un tema que como tal, es introducido al alumno en el sexto grado de primaria, por lo tanto, las actividades de éste grupo, integran el concepto de orden entre fracciones, permitiendo reafirmar las bases, e incrementar la habilidad del alumno para realizar comparaciones mentales complejas. Las actividades que componen a este grupo son las siguientes:

- Actividad básica del orden de fracciones. Con la finalidad de introducir al alumno el concepto de orden entre fracciones, ésta actividad gráfica se basa en la ubicación de fracciones en la recta numérica.
- Actividad intermedia del orden de fracciones. Mezcla representaciones gráficas y textuales que permiten al alumno, establecer la relación de orden entre fracciones, tales como indicar si una fracción es mayor, menor, o igual a otra.
- Actividad avanzada del orden de fracciones. Fomenta que el alumno practique el método numérico para determinar el orden de fracciones, mediante representaciones textuales de información.

2.2.3 Actividades de apoyo para el aprendizaje del grupo simplificación de fracciones

Debido a que la simplificación de fracciones es un caso particular de la equivalencia de fracciones, este grupo únicamente incorpora una actividad que busca agilizar los cálculos mentales del alumno sobre este tema. A continuación se describe la actividad:

- Actividad avanzada de la simplificación de fracciones. Promueve el empleo del método numérico para la simplificación de fracciones, brindando ejercicios que ponen en práctica los cálculos mentales del alumno.

En todas las actividades, el sistema genera una tarea y el usuario debe resolverla. La herramienta es responsable de verificar la entrada y retroalimentar al alumno sobre ésta.

El agrupamiento y clasificación de las actividades, brinda flexibilidad al sistema, permite que los estudiantes puedan buscar y seleccionar actividades arbitrariamente, y facilita que la herramienta se ajuste a las necesidades particulares de cada usuario. Estas características, sugieren el empleo de tecnología distribuida.

La herramienta se cataloga entonces, como una tecnología para la distribución de material de aprendizaje.

Capítulo 3

Arquitectura de distribución del sistema

En este capítulo se habla de algunas de las tecnologías de cómputo distribuido que existen y que pueden emplearse para implantar la herramienta, así también, se especifica la arquitectura de distribución del material en que se sustentará el desarrollo del sistema.

3.1 Tecnología a utilizar para la implementación de la propuesta

Las tecnologías de enseñanza más novedosas, tales como la distribución de material educativo, los laboratorios virtuales y las redes de educación, representan la implementación de aplicaciones distribuidas caracterizadas por elementos heterogéneos, dinámicos, y concurrentes que requieren de un alto grado de control [5]. Para hacer frente a la complejidad de dichos sistemas, puede emplearse la computación de objetos distribuidos, ya que ofrece las siguientes ventajas: reducción de costos de implementación, ubicación arbitraria de los clientes, acceso remoto y dinámico a los objetos distribuidos, etcétera [1].

La tecnología distribuida orientada a objetos brinda mejoras sustanciales a la productividad, así como al mejoramiento de la calidad, rendimiento, confiabilidad, e interoperabilidad de las aplicaciones de software.

3.1.1 Tecnologías distribuidas orientadas a objetos

Tres de los principales paradigmas de objetos distribuidos son: DCOM (Modelo de Objetos de Componentes Distribuidos - *Distributed Component Object Model*) de Microsoft, CORBA de OMG (Grupo Administrador de Objetos - *Object Management Group*) y Java RMI (Método de Invocación Remota - *Remote Method Invocation*) de JavaSoft.

Las arquitecturas DCOM, CORBA y Java/RMI proporcionan mecanismos de invocación transparente y acceso a objetos distribuidos remotos, en esencia las tres

tecnologías realizan funciones similares [2]. Debido a las perspectivas que se tienen del proyecto, se ha elegido a CORBA para su desarrollo, ya que a diferencia de DCOM cuenta con implementaciones libres y gratuitas, lo que para las escuelas primarias no implicará un costo de compra de software, además soporta un alto rango de lenguajes de programación, lo que facilita la integración de actividades desarrolladas en lenguajes de programación distintos, atributo del que carece la tecnología Java/RMI.

Actualmente existe una gran diversidad de implementaciones CORBA. Tras analizar y probar algunas de ellas, se optó por emplear MICO (*MICO is CORBA*) para el desarrollo de este proyecto, ya que es una alternativa libre, gratuita, completa y congruente del estándar CORBA. La implementación MICO trabaja con el lenguaje de programación C++, y es capaz de correr en múltiples plataformas. Por tener un perfil libre, la herramienta hace uso del sistema operativo Linux en su distribución Fedora Core 2.

3.1.2 Requerimientos tecnológicos

La plataforma de desarrollo y la tecnología elegida, imponen restricciones para el desarrollo del proyecto, debido a esto, para que las instituciones educativas o los alumnos puedan hacer uso de la herramienta, será necesario que dispongan de una computadora que cumpla con los requerimientos mínimos que le permitan compilar, instalar y ejecutar el software empleado. Estos requerimientos son: 256MB de memoria RAM, procesador de 1700 MHz, tarjeta de red y de video.

3.1.3 CORBA

En este apartado se comentan los elementos más relevantes de la arquitectura CORBA, un estudio completo de esta arquitectura puede encontrarse en [3].

El estándar CORBA aboga por el uso de sistemas abiertos, con interfaces estándar orientadas a objetos, construidos con hardware, redes, sistemas operativos y lenguajes de programación heterogéneos [6]. En congruencia, está diseñado bajo los siguientes principios: *Orientación a Objetos*, *Transparencia de locación*, *Neutralidad en el lenguaje de programación*, y *Soporte para puentes*.

Los elementos más importantes de la arquitectura CORBA son: El lenguaje IDL (Lenguaje de Definición de Interfaces - *Interface Definition Language*), la traducción de

IDL a lenguajes de programación (compiladores IDL) y la infraestructura de distribución de objetos ORB (Intermediario para la Solicitud de Objetos - *Object Request Broker*).
Enseguida se hablará del lenguaje IDL y de los compiladores IDL, el estudio del ORB se trata en la sección 3.2.

IDL. Lenguaje de Definición de Interfaces

La interfaz de los objetos CORBA se define usando el lenguaje de definición de interfaces IDL. El lenguaje IDL es puramente declarativo, no derivado de algún lenguaje de programación existente.

El definir las interfaces de los objetos distribuidos con un lenguaje homogéneo, hace posible que la tecnología CORBA soporte objetos implementados en lenguajes de programación heterogéneos. Los archivos IDL son compilados para generar código empleado en la implementación del cliente y del servidor.

Compiladores IDL

Los compiladores del lenguaje IDL son utilizados para mapear el código IDL a un lenguaje de programación específico. El compilador IDL, lee un archivo con definiciones en este lenguaje para producir archivos con código útil para la programación del cliente y del servidor. El código generado para el cliente recibe el nombre de código *stub*, mientras que el código del servidor es llamado *skeleton*.

El código *stub* y *skeleton*, es empleado para hacer a los clientes y servidores conscientes de las definiciones de los objetos que aparecen en el archivo IDL. El código *stub* es usado por el cliente para invocar operaciones en objetos remotos. El código *skeleton* es usado por el servidor para habilitar los servicios que implementan los objetos CORBA, y de ésta manera atender las solicitudes de los clientes.

El código *stub* y *skeleton* generalmente es utilizado cuando el cliente y el servidor funcionan de manera estática, sin embargo, pueden contener código necesario para que el cliente realice invocaciones dinámicas, o para que el servidor atienda las peticiones de los clientes en tiempo de ejecución.

3.2 Arquitectura de la herramienta propuesta

La figura 3.1 esquematiza la arquitectura de la herramienta, la cual está diseñada con base a la tecnología distribuida empleada. A continuación, se describen cada uno de los elementos que intervienen en el funcionamiento del sistema [1] [3] [18].

Cliente. El cliente es el medio que el usuario utiliza para comunicarse con el servidor, y es responsable de cargar, iniciar y dar fin a las actividades. Dentro del cliente se incluye la aplicación del usuario, la cual, permite interactuar con las actividades didácticas que residen en el servidor de manera transparente y amigable. Puesto que la herramienta funcionará en un ambiente cliente-servidor, el usuario será responsable de levantar al cliente, por lo tanto, será un requisito que el servidor esté ejecutándose. Para el adecuado funcionamiento del cliente, deben configurarse algunas variables de ambiente, conforme a lo señalado en al apéndice B.

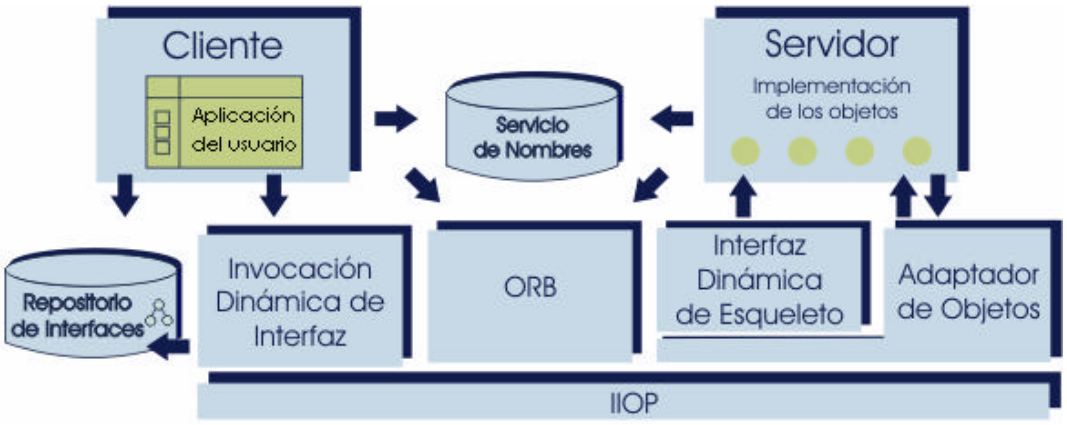

Figura 3.1. Arquitectura de la herramienta.

Repositorio de interfaces (IR - *Interface Repository*). Es un servicio *standalone* que actúa como un repositorio para todas las declaraciones IDL. Una vez que un archivo IDL ha sido dado de alta en el repositorio de interfaces, la descripción completa de las definiciones que aparecen en el archivo IDL están disponibles para aplicaciones CORBA que deseen usarlas.

Este servicio, es el complemento de las características dinámicas de CORBA. Cuando se usa la invocación dinámica o el esqueleto dinámico, es necesario que las aplicaciones encuentren las descripciones de las interfaces IDL en tiempo de ejecución. Esta información puede obtenerse haciendo invocaciones sobre el repositorio de interfaces.

Invocación dinámica de la interfaz (DII - *Dynamic Invocation Interface*). Esta parte de la arquitectura permite que el cliente invoque métodos de manera dinámica en tiempo de ejecución. El uso de la invocación dinámica, brinda flexibilidad al sistema, y facilita su futuro crecimiento.

Servicio de nombres CORBA. Es usado por los clientes para referenciar objetos remotos, y por los servidores para hacer públicos sus objetos (sirvientes) bajo un nombre propio. De esta manera, se permite a los clientes encontrar servicios y facilitar las conexiones iniciales con los sirvientes.
La manera en que se emplea este servicio, se muestra en la figura 3.2 [18][3], y se describe a continuación:

1. El servidor asocia el nombre del objeto con una referencia.
2. Si un cliente conoce el nombre de un objeto, puede solicitar la referencia del mismo al servicio de nombres.
3. Una vez que un cliente posee la referencia de un sirviente, puede hacer invocaciones remotas sobre las operaciones del objeto.

ORB. Es el bus de comunicación de los objetos. Permite a los objetos hacer solicitudes de manera transparente, así como recibir la respuesta de otros objetos locales o remotos. Los objetos CORBA nunca se comunican directamente entre sí, en vez de esto, el cliente solicita la información al ORB que corre en la máquina local. El ORB local transmite la solicitud al ORB de la máquina remota. El ORB remoto localiza al objeto servidor apropiado y retorna una referencia al objeto que lo solicitó.

[3] Orfali, R., Harhey, D. *Client/Server Programming with Java and CORBA*. Wiley Computer Publishing, 1998.

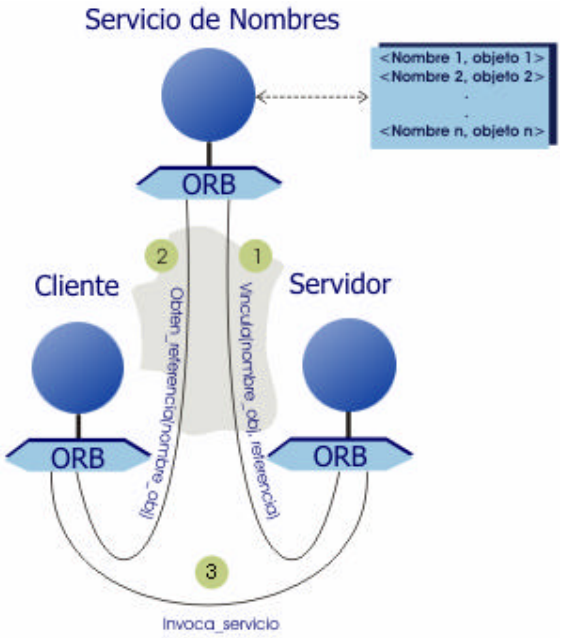

Figura 3.2. Interacción del cliente y del servidor con el Servicio de Nombres.

Un ORB CORBA proporciona una variedad extensa de servicios *middleware* . EL ORB permite a los objetos descubrirse unos a otros en tiempo de ejecución para invocar sus servicios. Algunas de las características que hacen a un ORB superior a otras opciones middleware son:

- Métodos de invocación estática y dinámica. Permite definir estáticamente las invocaciones a métodos en tiempo de compilación, o descubrirlos dinámicamente en tiempo de ejecución.
- Vinculación con lenguajes de alto nivel. Es capaz de invocar métodos de objetos sirvientes usando un lenguaje de alto nivel arbitrario y elegido por el programador.
- Sistemas auto descriptivos. Proporciona metadatos en tiempo de ejecución que describen la interfaz de los objetos sirvientes.
- Transparencia local o remota. Puede correr en modo *standalone*, o puede estar interconectado con otros ORB's.
- Seguridad y transacciones. Incluye información sobre el contexto de la comunicación de los objetos en sus mensajes. Esto permite el manejo de seguridad y transacciones a través de las máquinas y de los límites del ORB.
- Mensajería Polimórfica. Invoca funciones remotas en objetos específicos. Esto significa que el llamado a la función tiene diferentes efectos dependiendo del

objeto que lo reciba.

- Coexistencia con sistemas existentes. La separación que hace CORBA de la definición de los objetos y de sus implementaciones, facilita encapsular aplicaciones existentes. Usando el IDL de CORBA, es posible hacer lucir el código de aplicaciones existentes como un objeto más en el ORB.

Servidor. La función primordial del servidor es mantener el repositorio de actividades didácticas, objetos CORBA, que integran el sistema, y que están disponibles al cliente. El servidor consta de una aplicación que permite que distintos clientes se interconecten y accedan a los objetos necesarios de forma concurrente. Para el funcionamiento adecuado del servidor, es necesario configurar algunas variables de ambiente, la configuración necesaria puede consultarse en el apéndice B.

Interfaz dinámica de esqueleto (DSI - *Dynamic Skeleton Interface*). Proporciona un mecanismo en tiempo de ejecución que permite que el servidor manipule llamadas a métodos. Analiza los valores de los parámetros de las llamadas remotas para determinar el objeto y método al que van dirigidas. La interfaz de esqueleto dinámica es el equivalente de la invocación dinámica de interfaces pero desde la perspectiva del servidor.

Adaptador de Objetos. La tarea del adaptador de objetos, es hacer a los objetos CORBA accesibles a los clientes por medio de la red. La responsabilidad primaria de un adaptador de objetos es asegurarse que una invocación, sea local o remota, se transmita apropiadamente al objeto destino.
El ORB que implementa MICO hace uso del adaptador de objetos POA (Adaptador de Objetos Portable -*Portable Object Adapter*). El adaptador POA es estándar y está descrito en la especificación de CORBA. Por lo tanto, el código de implementación que haga uso del adaptador POA, es portable a distintas implementaciones de ORB's.

IIOP (Protocolo Internet Inter-ORB / *Protocolo Internet Inter.-ORB*). Es un protocolo de comunicación diseñado por la OMG para facilitar la transferencia de solicitudes CORBA de un ORB a otro. Especifica como se deben intercambiar mensajes en una red TCP/IP (Protocolo de Control de Transmisión/Protocolo de Internet – Transfer Control Protocol/Internet Protocol).

Capítulo 4

Modelado de la arquitectura del sistema

El presente capítulo documenta el modelado de la arquitectura funcional del sistema, con lo cual se cubre parcialmente el cuarto objetivo específico de esta propuesta, la parte restante, el desarrollo de la herramienta, se aborda en el siguiente capítulo.

El presente proyecto ha sido modelado bajo el estándar UML (Lenguaje Unificado de Modelado - *Unified Modeling Language*) siguiendo un proceso de desarrollo iterativo. Enseguida se presentan los modelos de casos de uso y los diagramas de clase y de secuencia generados para el proyecto.

4.1 Modelo de casos de uso

Los casos de uso, son documentos narrativos que describen la secuencia de eventos que un usuario ejerce al sistema para completar un proceso. Los casos de uso identificados en el sistema se describen en las secciones 4.1.1 y 4.1.2.

4.1.1 Caso de uso: Inicia Actividad

Es deseable que el diseño de laherramienta permita que los usuarios, en especial los alumnos, trabajen a su propio ritmo y elijan de manera arbitraria las actividades que desean manipular en un momento determinado, por tal motivo, el sistema debe ser flexible y establecer comunicación únicamente con los objetos relacionados con las actividades que están siendo utilizadas por los usuarios, y no con todos los objetos existentes en el servidor. Con la finalidad de documentar la funcionalidad del sistema en el caso antes descrito, surge el caso de uso “Inica Actividad”, el cual se describe en la tabla 4.1.

Tabla 4.1. Caso de uso Inicia Actividad.

Actores:	Usuario.
Descripción:	Este caso de uso se presenta cuando el usuario debe elegir una actividad con la cual trabajar.
Tipo:	Primario**.**
Precondiciones	La aplicación de usuario está ejecutandose .
Flujo normal de eventos.	

Acción del actor	Respuesta del sistema
1. Inicia cuando el usuario debe elegir una actividad para trabajar con ella.	2. El sistema muestra los grupos de actividades.
3. El usuario elige un grupo de actividades.	4. El sistema muestra las actividades pertenecientes al grupo elegido.
5. El usuario elige una actividad.	6. El sistema obtiene una referencia al objeto CORBA relacionado con la actividad, mediante el servicio de nombres CORBA.

Flujos alternos.	Excepción del sistema: El sistema muestra un mensaje de error al usuario y le presenta las opciones que se pueden tomar para reestablecer la aplicación.
Poscondiciones.	La actividad comienza a ejecutarse.

4.1.2 Caso de uso: Trabaja Actividad

Una vez que el usuario ha dado inicio a alguna de las actividades, comienza el principal trabajo del sistema, que es responder a las entradas del usuario mediante llamadas dinámicas a funciones de objetos CORBA que residen en el servidor. Para documentar de mejor manera la forma en que el sistema opera, se presenta el caso de uso “Trabaja Actividad”, el cual se describe en la tabla 4.2.

Los casos de uso anteriormente descritos, guardan una estrecha relación, la cual se representa en el diagrama de casos de uso de la figura 4.1.

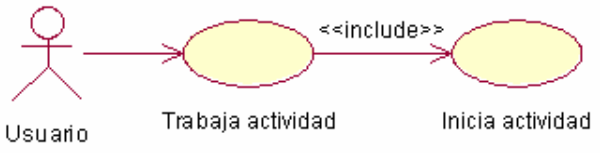

Figura 4.1. Diagrama de casos de uso.

Tabla 4.2. Caso de uso Trabaja Actividad.

Actores:	Usuario.
Descripción:	Este caso de uso se presenta cuando el usuario comienza a trabajar con alguna de las actividades que dispone el sistema.
Tipo:	Primario**.**
Precondiciones	Se ha iniciado una actividad.

Flujo normal de eventos.	
Acción del actor	Respuesta del sistema
1. Comienza cuando el usuario ha elegido una actividad, y se dispone a trabajar con ella.	2. El sistema muestra la actividad al usuario.
3. El usuario interactúa con la actividad indicándole la operación remota por ejecutar.	4. El sistema accede al repositorio de interfaces para invocar dinámicamente la operación apropiada del objeto referenciado, en caso de error se ejecuta el flujo alterno excepción del sistema.
5. El usuario finaliza la actividad.	

Flujos alternos.	Excepción del sistema: El sistema muestra un mensaje de error al usuario y le presenta las opciones que se pueden tomar para reestablecer la aplicación.
Poscondiciones.	Finaliza la actividad.

4.2 Diagrama de Clases de la Interfaz IDL

En esta sección, se exponen las clases que intervienen en el flujo de eventos de los casos de uso. Debido a que la herramienta funciona de manera distribuida, fue necesario diseñar antes que nada las interfaces de los objetos CORBA que son empleados por el cliente.

Los diagramas de clase de la interfaz IDL han sido modelados en base a la especificación “Rose CORBA” de Rational, la cual forma parte de Rational Rose 98i. Esta especificación trabaja con el lenguaje de modelado UML. La especificación UML para CORBA, fue diseñada para proveer un significado estándar a la semántica del lenguaje IDL empleado por CORBA [17].

Enseguida se describen los diagramas de clase generados para la interfaz IDL de la herramienta.

4.2.1 Módulo del sistema

El servidor debe ofrecer objetos que encapsulen métodos dirigidos a manipular fracciones a nivel de sexto grado de primaria. Debido a que el sistema crecerá en el futuro, es necesario agrupar los objetos relacionados bajo el mismo alcance, lo cual se logra al declarar las interfaces de los objetos en un mismo módulo que las contenga. El módulo que contiene las declaraciones de las interfaces, tipos de datos, etc., recibe el nombre de Fracciones_Sexto y se encuentra representado en la figura 4.2.

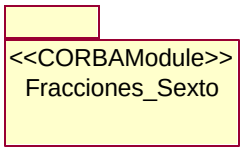

Figura 4.2. Módulo Fracciones_Sexto.

4.2.2 Estructuras

Debido a que el cliente debe ser capaz de invocar operaciones del servidor de manera dinámica, se requiere que en tiempo de ejecución se consulte el repositorio de interfaces para descubrir la sintaxis de las operaciones que se desean invocar. Si las operaciones invocadas, reciben dentro de sus parámetros datos definidos por el usuario, la estructura de estos datos debe ser conocida tanto por el cliente como por el servidor, y por lo tanto es necesario definir los datos en la interfaz IDL.

Como se ha mencionado con anterioridad, los objetos servidor manipulan fracciones en sus operaciones, por lo tanto, el tipo de dato fracción debe ser declarado en la interfaz IDL. Este tipo de dato, modelado en la figura 4.3, consta de una estructura cuyos elementos son el numerador, denominador y una bandera que indica el signo de la fracción.

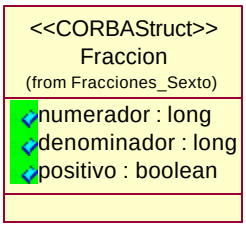

Figura 4.3. Tipo de dato fracción.

4.2.3 Interfaces

Las definiciones de las interfaces de los objetos que implementa el servidor, se muestra en la figura 4.4. Cada una de estas interfaces proporciona operaciones dirigidas a manipular fracciones en algún subtema específico de las matemáticas de sexto grado.

La descripción de las interfaces, se presentan a continuación de forma breve:

- Interfaz Crea_Fraccion. Contiene operaciones necesarias para crear fracciones. Esta interfaz es heredada por el resto de las interfaces.
- Interfaz Equivalencia. Permite determinar relaciones de equivalencia entre un par de fracciones.
- Interfaz Orden. Se integra de métodos para obtener la relación de orden entre fracciones.
- Interfaz Simplificación. Proporciona funciones para simplificar fracciones.

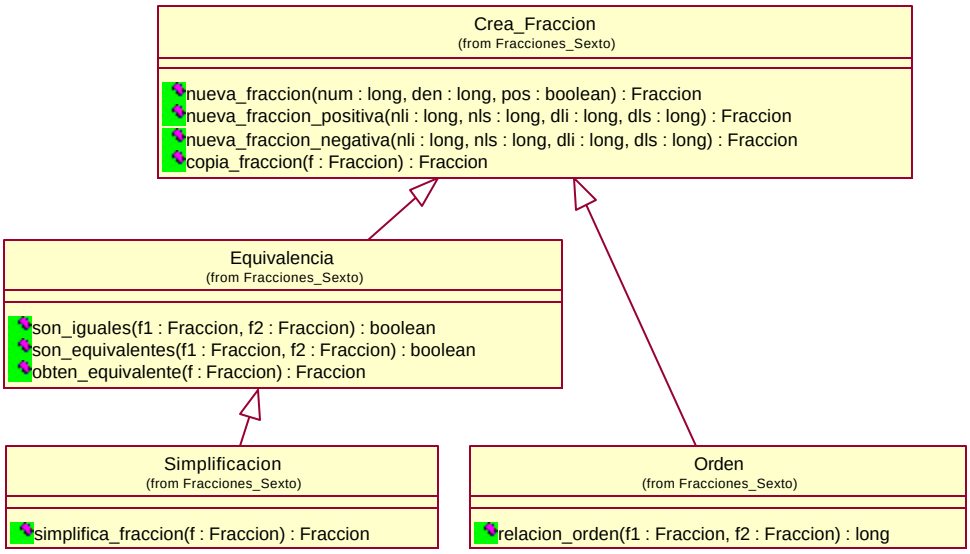

Figura 4.4. Interfaces de los objetos servidor.

4.3 Diagramas de clase

Cada interfaz IDL está relacionada con una clase en el servidor que la implementa. Las operaciones de las clases del servidor, dan soporte a las operaciones de la interfaz asociada, además, estas clases contienen los métodos *invoke()* y *_primary_interface()* que hacen posible el uso de la interfaz dinámica del esqueleto. La relación entre las clases del servidor, se ilustra en el diagrama de la figura 4.5.

Respecto a las clases que residen en el cliente, las cuales se muestran en la figura 4.6, se puede mencionar que la clase *PrincipalInterfaz*, representa la interfaz principal del sistema, y es con la que el usuario debe interactuar al inicio de la aplicación. La clase *Grupos*, se diseñó para permitir elegir al usuario un grupo de actividades, ya sea orden, equivalencia o simplificación. Las clases *Orden*, *Equivalencia*, y *Simplificación*, permiten al usuario elegir algunas de las actividades de estos grupos. Las clases *Act_orden_01*, *Act_orden_02*, *Act_orden_03*, *Act_equiv_01*, *Act_equiv_02* y *Act_simpl_01* representan la interfaz de cada una de las actividades. Por último, la clase *Control_actividad*, encapsula el funcionamiento principal que se ejerce en el cliente, como lo es la invocación dinámica de interfaces.

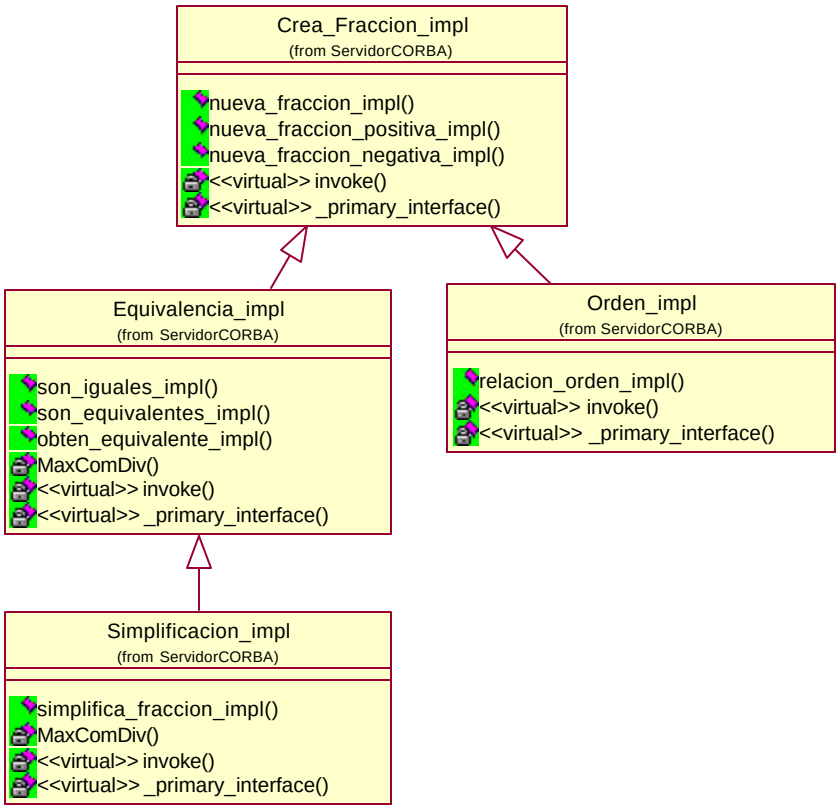

Figura 4.5. Diagrama de clases del servidor.

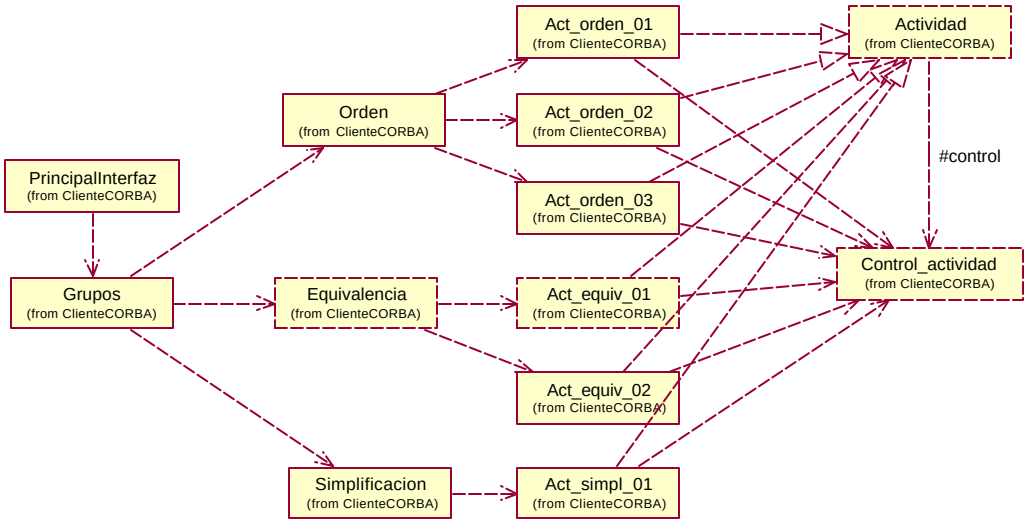

Figura 4.6. Diagrama de clases del cliente.

4.4 Diagramas de secuencia

En función de las clases propuestas, se han diseñado los diagramas de secuencia para los casos de uso existentes. Enseguida se muestran los diagramas de secuencia asociados con una actividad particular, sin embargo, estos diagramas pueden generalizarse al resto de las actividades, pues todas funcionan de manera muy similar.

El diagrama de secuencia correspondiente al caso de uso “Inicia Actividad”, se ilustra en la figura 4.7, y ejemplifica las tareas del sistema durante la inicialización de una actividad del grupo orden.

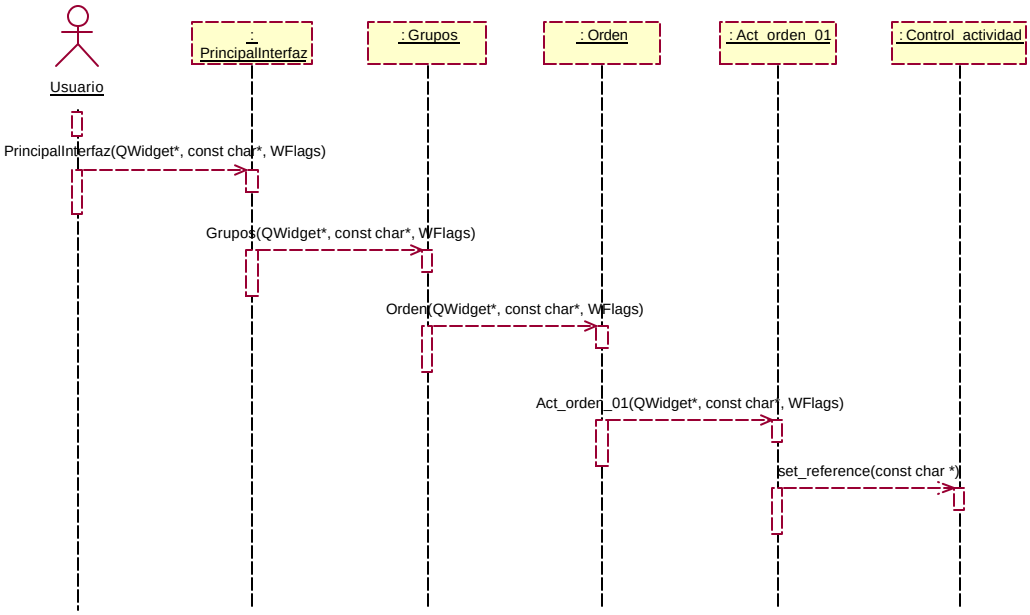

Figura 4.7. Diagrama de secuencia de inicia actividad.

La figura 4.8 representa el diagrama de secuencia del caso de uso “Trabaja Actividad”, este diagrama esta asociado con la actividad orden 01.

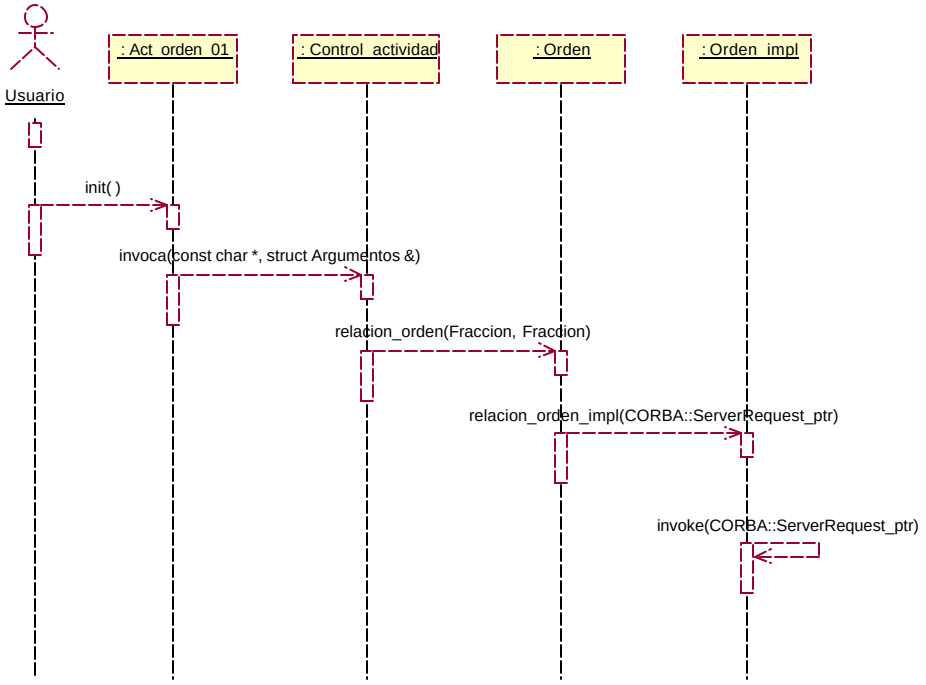

Figura 4.8. Diagrama de secuencia de trabaja actividad.

Capítulo 5

Implementación del sistema

Este capítulo documenta los aspectos más relevantes de la implementación del sistema. Con lo que aquí se presenta, se concluye el cuarto objetivo específico iniciado en el capítulo anterior.

5.1 Flujo de la implementación del sistema

En la figura 5.1 se muestran los pasos más importantes del flujo de implementación del sistema.

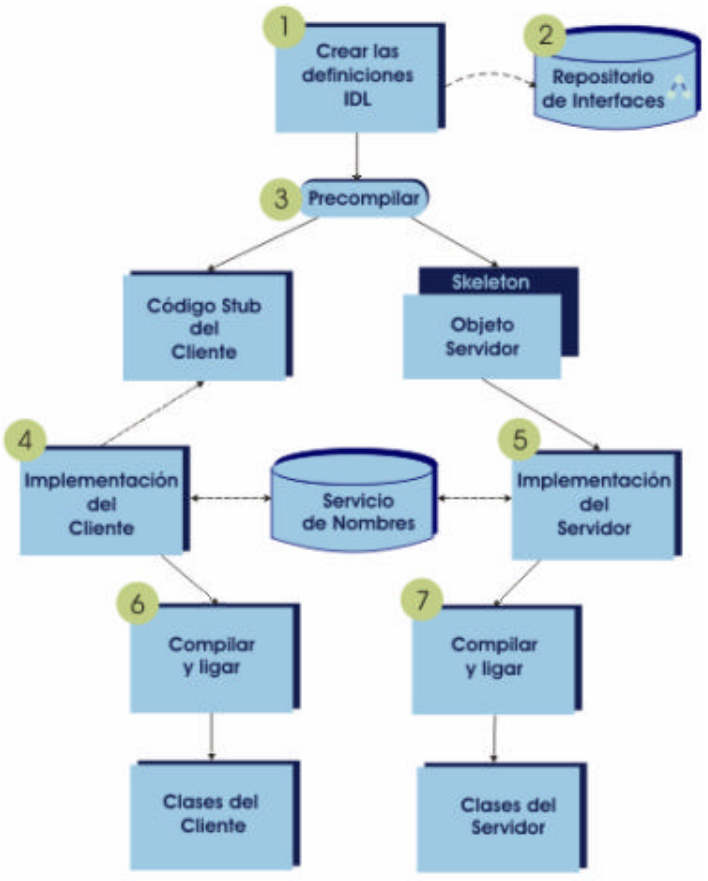

Figura 5.1. Flujo de implementación del sistema.

Para el mejor entendimiento del proceso de implementación, en las siguientes secciones se hablará de cada uno de los pasos que fueron identificados, los cuales consisten en:

1. Crear las definiciones IDL.
2. Alimentar el repositorio de interfaces.
3. Precompilar el archivo IDL.
4. Implementar el cliente.
5. Implementar el servidor.
6. Compilar los archivos del cliente.
7. Compilar los archivos del servidor.

5.1.1 Crear las definiciones IDL

Con base al modelado de la arquitectura del sistema, se definieron los módulos, estructuras, e interfaces de los objetos que estarían disponibles en el servidor. Tales definiciones se almacenaron en el archivo de nombre *fraccion.idl*. El contenido de este archivo se muestra en el apéndice C.

5.1.2 Repositorio de interfaces

El siguiente paso consiste en hacer que las definiciones contenidas en el archivo IDL estén disponibles para los clientes, para esto es necesario alimentar el repositorio de interfaces con el contenido del archivo *fraccion.idl*.

Antes de alimentar el repositorio de interfaces, se necesita levantar al demonio que instancia este servicio. MICO permite levantar este demonio con el siguiente comando:

```
ird -ORBIIOPAddr <direccion_repositorio>
```

La opción –ORBIIOPAddr define la dirección de Internet en la que debe correr el servicio. Las direcciones de Internet se definen como cadenas de texto que cumplen el siguiente formato:

```
inet:<nombre del host>:<número de puerto>
```

Una vez levantado el *ird*, se alimenta el repositorio de interfaces con las definiciones IDL. MICO proporciona una herramienta que funciona en línea de

comandos y que realiza ésta labor. Esta herramienta es conocida como *idl*. La herramienta *idl* es usada para mapear código IDL a C++, así como para alimentar el repositorio de interfaces con especificaciones IDL.

El comando empleado para alimentar el repositorio de interfaces se muestra a continuación:

```
idl --no-codegen-c++ --feed-ir -ORBIfaceRepoAddr <direccion_repositorio> fraccion.idl
```

La opción *–no-codegen-c++* deshabilita la generación de código C++, mientras que la opción *–feed-ir* indica que el archivo *fraccion.idl* debe ser alimentado en el repositorio. La opción *–ORBIfaceRepoAddr* especifica la dirección de Internet del proceso que corre la implementación del repositorio de interfaces.

5.1.3 Precompilar

Continuando con el flujo de implementación, se precompila el archivo IDL para generar el código *stub* y *skeleton*.

Nuevamente, para mapear el código IDL a C++ se utiliza la herramienta *idl*. El comando empleado se muestra a continuación:

```
idl –any fraccion.idl
```

La opción *–any* activa el soporte para la inserción y extracción de operadores definidos por el usuario hacia o desde un tipo de dato *any*.

5.1.4 Implementación del cliente

En este apartado se describe el mecanismo empleado por el cliente para obtener referencias a objetos remotos, así como para invocar interfaces dinámicamente.

Obtención de referencias a objetos remotos

Cuando el usuario elige una actividad, la aplicación cliente debe obtener una referencia al objeto sirviente mediante el servicio de nombres CORBA. La clase *Control_actividad* se encarga de llevar a cabo esta labor dentro del método *set_reference*.

Los pasos a realizar para obtener una referencia a un objeto, son: referenciar al servicio de nombres, posicionarse en la raíz del árbol que almacena los nombres, y

solicitar la referencia del objeto deseado. El código que implementa estos pasos se encuentra en el apéndice C.

Invocación Dinámica de Interfaces

La Invocación Dinámica de Interfaces es una tarea compleja. Con la finalidad de hacer entendible el mecanismo empleado para realizar esta labor, en la figura 5.2 se esquematiza los pasos que se deben llevar a cabo para invocar una operación remota.

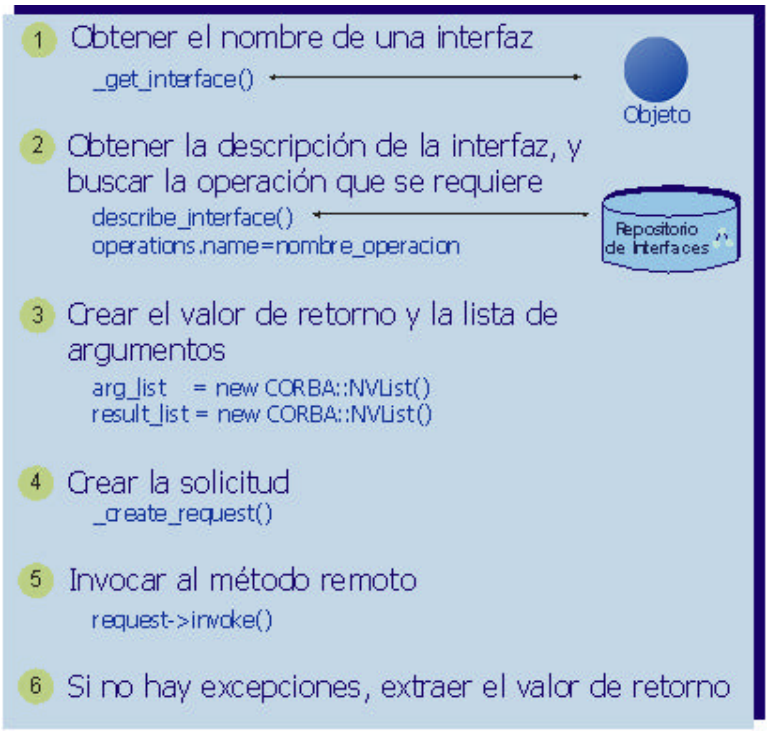

Figura 5.2. Flujo de la Invocación Dinámica de Interfaces.

A continuación se da una descripción de cada uno de estos pasos:

1. *Obtener el nombre de la interfaz*. Una vez que se obtiene la referencia del objeto sirviente, se puede obtener el nombre de la interfaz del sirviente invocando el método *_get_interface* sobre su referencia. Esta invocación retorna una referencia del objeto *InterfaceDef*. *InterfaceDef* es un objeto dentro del repositorio de interfaces que describe la interfaz del objeto servidor.
2. *Obtener la descripción de la interfaz, y buscar la operación que se requiere*. Puede emplearse el objeto *InterfaceDef* como punto de entrada para navegar en el repositorio de interfaces. Dentro del repositorio, es posible obtener toda clase de información detallada acerca de la interfaz y de los métodos que

soporta. En este punto se emplea el método *describe_interface* para obtener una descripción completa de la interfaz, y posteriormente encontrar el método que se desea invocar.

3. *Crear el valor de retorno y la lista de argumentos*. CORBA especifica una estructura de datos autodescriptiva para pasar parámetros, esta estructura se conoce como *NVList*. Tanto el valor de retorno como la lista de argumentos que recibe la operación remota pueden crearse con el método *create_list*. Una vez creada la lista, se llama a la operación *add_value* las veces que sean necesarias para agregar elementos a la lista.
4. *Crear la solicitud*. La solicitud de la operación remota se crea invocando al método *_create_request*. La solicitud contiene el nombre de la operación a invocar, la lista de argumentos y el valor de retorno.
5. *Invocar el método remoto*. Creada la solicitud, lo que resta es invocarla. El método *invoke* envía la solicitud y obtiene los resultados.
6. *Si no hay excepciones, extraer el valor de retorno*. El sistema fue diseñado para que los valores retornados por las operaciones remotas, se almacenaran en anys. Para poder extraer el valor contenido en el *any* hay que identificar el tipo de dato que éste contiene y almacenarlo en una variable adecuada.

La implementación de las invocaciones dinámicas es realizada por la operación *invoca* de la clase *Control_actividad*. El código de esta operación se presenta en el apéndice C.

5.1.5 Implementación del servidor

En esta sección se expone la forma en que el servidor realiza sus tareas. Además, se desglosa el proceso de desarrollo que debe seguir un sirviente dinámico para atender las solicitudes.

5.1.5.1 La aplicación servidor

Cuando la aplicación servidor inicia, debe obtener referencias a un conjunto básico de objetos iniciales, por ejemplo al ORB, al adaptador de objetos, y al servicio de nombres. El estándar CORBA define el servicio de inicialización para encargarse de esta tarea.

Comúnmente el servicio de inicialización es empleado para obtener una referencia inicial al contexto raíz del servicio de nombres. Una vez que se dispone de una

referencia al contexto inicial, puede obtenerse referencias a otros servicios CORBA. Finalmente, el servidor debe activar los objetos sirvientes y al POA.

Los pasos que una aplicación servidor sigue durante la fase de inicialización se ilustran en la figura 5.3.

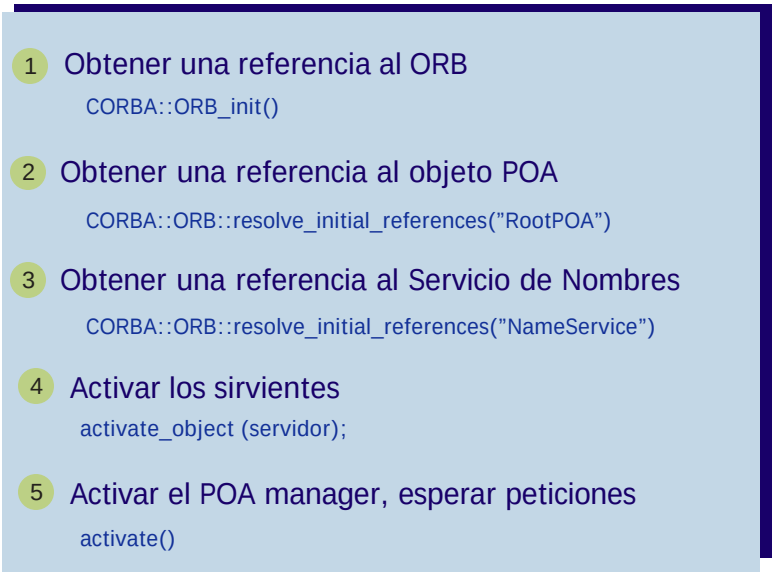

Figura 5.3. Inicialización del servidor.

A continuación, se dará una explicación de estos pasos:

1. Obtener una referencia al ORB. Para obtener una referencia al ORB, se invoca la función *CORBA::ORB_init()*. Esta función retorna un objeto de tipo CORBA::ORB. En el llamado al método, se pasan argumentos que configuran algunas propiedades relacionadas con el ORB.
2. Obtener una referencia al POA. Para obtener una referencia al POA, se invoca la operación *resolve_initial_* references del objeto ORB obtenido en el paso anterior. Puesto que *resolve_initial_references* se emplea para obtener referencias a distintos servicios, debe invocarse con el paráme tro RootPOA. El llamado retorna un objeto de la clase CORBA que debe ser convertido a uno de la clase PortableServer::POA mediante la función _narrow().
3. Obtener una referencia al objeto Servicio de Nombres. La referencia al servicio de nombres se obtiene invocando la operación *resolve_initial_references* con parámetro NameService. El objeto CORBA que devuelve esta función debe convertirse a uno de la clase CosNaming::NamingContextExt.

4. Activar los sirvientes. En esta etapa, el servidor crea instancias de los sirvientes y los activa invocando la operación *activate_object()* del objeto POA obtenido en el paso 2. Posteriormente, el servidor publica los sirvientes en el servicio de nombres empleando la referencia de éste servicio que se obtuvo en el paso anterior.

   Dentro del servicio de nombres, las relaciones entre el nombre del objeto y su referencia, se organizan en una jerarquía de árbol. Existen dos clases de relaciones en la jerarquía: Las relaciones de contexto y las relaciones de objeto. Las primeras son una asociación entre un nombre y un contexto, éste tipo de relaciones puede visualizarse como los directorios de un sistema de archivos. Mientras que las segundas son asociaciones entre un nombre y una referencia al objeto, éstas relaciones son las hojas del árbol.

   Cuando se da a conocer un sirviente en el servicio de nombres, hay que determinar, si existe el contexto al que pertenecerá, en caso de que no exista debe crearse.
5. Activar al POA Manager y esperar peticiones. Una vez que se han levantado los sirvientes, hay que activar al POA y ejecutar al ORB. Esto se logra invocando la operación *activate()* del objeto POA, y la operación *run()* del objeto ORB.

El código que implementa la aplicación servidor puede consultarse en el apéndice C.

5.1.5.2 Los objetos sirvientes

Enseguida se presenta el proceso de implementación respetado para los sirvientes DSI. El proceso se ejemplifica con la clase *Crea_Fraccion_Impl*, sin embargo, puede generalizarse para el resto de los objetos.

Un sirviente dinámico se implementa al definir una nueva clase que hereda del *PortableServer::DynamicImplementation*. La clase *PortableServer::DynamicImplementation* define algunos métodos abstractos, entre ellos: *primary_interface e invoke*.

El método primary_interface

Una de las primeras cosas que un sirviente dinámico tiene que hacer al recibir una solicitud, es determinar, que clase de interfaz se está invocando. Puesto que las solicitudes recibidas no encapsulan información acerca de la interfaz a la que van dirigidas, la tarea de determinar el objeto destino se delega al ORB.

Para que el ORB encamine las solicitudes al objeto adecuado, debe conocer el tipo de los objetos sirvientes. El ORB conoce el tipo de los sirvientes, al invocar en ellos la función *primary_interface* . La implementación que los sirvientes hacen de la función *_primary_interface,* retorna un identificador de la interfaz del objeto.
El código que implementa esta función, se encuentra en el apéndice C.

El método invoke

El método *invoke* es llamado cada que se realiza una invocación en la implementación dinámica del objeto. El método recibe un parámetro de tipo *CORBA::ServerRequest*. Dentro de la implementación del método *invoke* se analiza la solicitud para determinar que operación ha sido invocada, y así poder atenderla apropiadamente.

La figura 5.4 muestra los pasos a seguir para atender una invocación dinámicamente. Enseguida se comentan estos pasos.

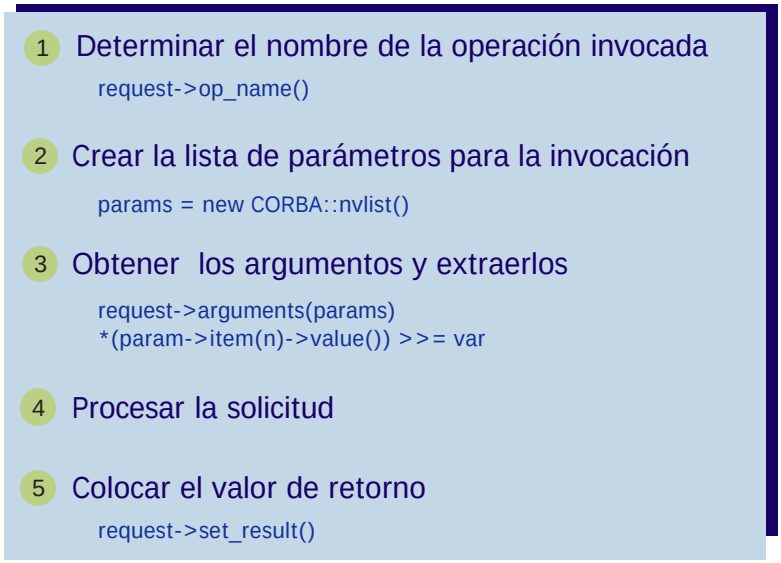

Figura 5.4. Flujo de la Invocación Dinámica de Esqueletos.

1. *Determinar el nombre de la operación invocada*. La función *invoke* debe ser capaz de determinar el nombre de la operación que ha sido invocada por el cliente. Esto se logra llamando a la función *op_name* de la solicitud. Identificada la operación que ha sido invocada, debe proveerse un mecanismo para asignarle el procesamiento de los parámetros. El apéndice C muestra la implementación de esta función.
2. *Crear la lista de parámetros para la invocación*. Una vez que los parámetros se han encaminado a la operación correspondiente, el primer paso antes de su procesamiento, es almacenarlos en una variable del tipo NVList. Por lo tanto, en este paso se crea la variable que los contendrá.
3. *Obtener los argumentos y extraerlos*. Una llamada al método arguments(NVList) extrae los valores de los argumentos de la solicitud, y los coloca en la variable NVList listos para ser procesados.
4. *Procesar la solicitud*. Se procesan los parámetros para producir un resultado.
5. *Colocar el valor de retorno*. Si se requiere retornar el resultado del procesamiento, se debe invoca r a la función *set_result*.

El código que implementa los pasos 2 a 5 se muestra en el apéndice C, dicho código pertenece a la operación *nueva_fracción_positiva de la clase Cra_Fracción_Impl*.

5.1.6 Compilar y ligar los archivos del cliente

MICO proporciona dos scripts dirigidos a simplificar la tarea de compilación y ligado de los archivos. Los scripts hacen uso del compilador c++ junto con algunas opciones de compilación que optimizan la aplicación resultante.

Enseguida se ejemplifica la forma en que deben compilarse los archivos:

Compilación.	mico-c++ -I –c codigo_cliente.cc –o codigo_cliente.o
Ligado.	mico-ld –o cliente codigo_cliente.o –lmico2.3.11

La opción *–c* de la compilación, le indica al compilador que debe compilar los archivos sin ligarlos, y la opción *–o* indica el archivo de salida.

La opción –lmico2.3.11 del ligado, establece que debe emplearse la biblioteca de mico.

Para obtener mayor información sobre la forma de compilación, puede consultarse el manual de referencia que se proporciona con mico.

5.1.7 Compilar y ligar los archivos del servidor

La forma en que se compilan y ligan los archivos del servidor es similar a lo realizado con los archivos del cliente.

Capítulo 6

Modelado de la interfaz de usuario

La herramienta que se ha gestado, será empleada principalmente por niños entre los diez y doce años de edad, debido a esto, el diseño de la interfaz es de vital importancia. El diseño debe considerar las necesidades particulares de los niños, proporcionándoles una herramienta intuitiva, de fácil uso y rápida adaptación. Además, los profesores podrían hacer uso del sistema si se dan a la tarea de dirigir a sus alumnos a lo largo de las actividades, por lo que la misma interfaz deberá ser útil para maestros y alumnos.

Con el afán de tomar en consideración los aspectos mencionados, y dando seguimiento al cuarto objetivo específico, se ha modelado la interfaz de usuario bajo un diseño centrado en el uso [11]. Esta clase de diseño parte de la filosofía de diseño centrado en el usuario, pero pone especial atención a los aspectos particulares de los usuarios que son más relevantes para el diseño de la interfaz, con lo que se asegura que la funcionalidad de la aplicación es correcta y que las necesidades del usuario serán satisfechas.

El diseño centrado en el uso se compone de tres modelos abstractos: *Modelo de roles de usuarios*, *Modelo de casos de uso*, y *Modelo de contenido*. Cada uno de estos modelos, se integra de una colección de descripciones y de un mapa de relaciones de estas descripciones. A continuación se muestran estos modelos para la interfaz de la herramienta.

6.1 Modelo de roles de usuario

El modelo de roles de usuario es una lista de relaciones que los usuarios pueden tomar al interactuar con el sistema. Un rol de usuario se representa por un nombre y una descripción breve de características sobresalientes de ese rol. Los roles que los usuarios pueden tomar al usar la herramienta, se presentan en la figura 6.1. La figura 6.2, ilustra las relaciones entre los roles identificados, e indica que, el rol profesor es una extensión del rol alumno.

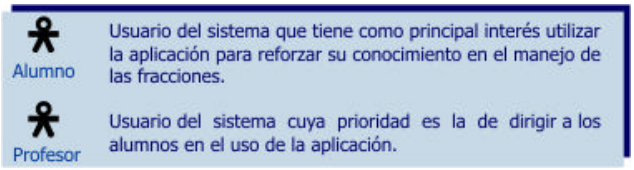

Figura 6.1. Roles de usuario.

Figura 6.2. Mapa de relaciones de los roles de usuario.

6.2 Modelo de casos de uso

Los casos de uso modelan la estructura esencial de las tareas, se diseñan con un nivel de abstracción que no oculta o anticipa suposiciones acerca de los detalles de la interfaz. De este modo se promueve la innovación y la creatividad de las interfaces [11].

Los casos de uso que se han identificado y modelado, se presentan ampliamente en la sección D.1 del apéndice D, a continuación se hablará de ellos brevemente:

- Eligiendo un grupo de actividades. Presenta la interacción entre la interfaz de la herramienta y el usuario para elegir un grupo de actividades.
- Eligiendo una actividad. Este caso, muestra las acciones que el usuario ejerce con el sistema cuando elige una actividad.
- Trabajando con la actividad. Da a conocer las tareas que el usuario debe realizar, al trabajar con una actividad.
- Reportando una excepción fatal. Manifiesta la interacción entre la interfaz y el usuario cuando se reporta una excepción fatal.

La figura 6.3 presenta el mapa de relaciones entre los casos, y hace notar que, para que un profesor o alumno pueda trabajar con una actividad, se requiere elegir primeramente la actividad dentro de un grupo de actividades. Además, muestra que en cualquier punto del sistema se puede reportar una excepción fatal que podría ser causada por algún evento fuera del control del sistema, como lo es un error en la red o en la tecnología distribuida empleada.

6.3 Modelo de contenido

Este modelo representa mediante prototipos abstractos los contenidos de la interfaz de usuario, sin atender a los detalles de como lucirá o se comportará la misma [10]. En un prototipo abstracto, la interfaz es una colección de materiales, herramientas, y áreas de trabajo que se describen en términos de la función que

proporcionan, sus objetivos, o usos. Los materiales son los objetos que los usuarios están interesados en manipular u observar, las herramientas son aquellas que permiten a los usuarios manipular los materiales, y los espacios de trabajo son las diversas partes de la interfaz que combinan las herramientas y materiales en una colección útil para el usuario.

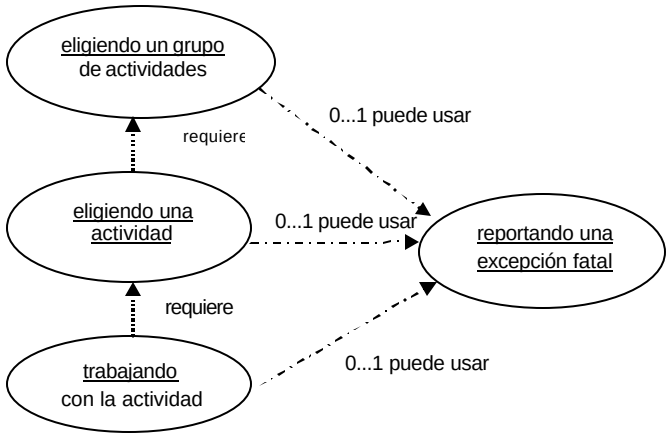

Figura 6.3. Mapa de relaciones de los casos de uso.

Los prototipos abstractos de la herramienta son:

- *Prototipo abstracto para el caso de uso "Eligiendo un grupo de actividades".* Aquí, se identificaron los siguientes materiales: Contenedor de grupo de actividades, e Informador de grupo. Las herramientas de este prototipo son: Seleccionador de grupo de actividad, y Terminador de aplicación.
- *Prototipo abstracto para el caso de uso "Eligiendo una actividad".* Los materiales identificados son: Contenedor de actividades, e Informador de actividad. Sus herramientas son: Seleccionador de actividad, y Retorno a grupo de actividades.
- *Prototipo abstracto para el caso de uso "Trabajando con la actividad".* Los materiales que fueron identificados son: Actividad, y Estado de la actividad. Sus herramientas son: Retorno a eligiendo una actividad, y Comunicador.
- *Prototipo abstracto para el caso de uso "*Reportando una excepción fatal*".* El material que se identificó es: Excepción. Su herramienta es: Seleccionador de opción.

En la sección D.2 del apéndice D puede obtenerse mayor información sobre estos prototipos.

El mapa de navegación de las diversas áreas de trabajo se encuentra representado en la figura 6.5, e ilustra los posibles caminos que el usuario tiene para navegar entre las pantallas de la interfaz, para comprender este mapa, es necesario hacer referencia a la simbología mostrada en la figura 6.4.

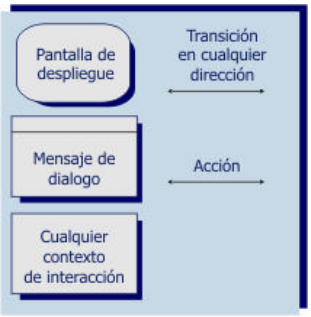

Figura 6.4. Simbología empleada en la representación del mapa de navegación de las áreas de trabajo.

Figura 6.5. Mapa de navegación de las áreas de trabajo.

6.4 Mapeo del modelado de la interfaz a su apariencia externa

Finalizado el modelado de la interfaz, se emprendió la tarea de mapearlo a elementos definidos, para lo cual fue necesario considerar las habilidades físicas y cognitivas de los niños, con objeto de conocer el estilo de interacción que debería proveer la herramienta.

De acuerdo con algunos estudios [4], los niños que se encuentran entre los 10 y 12 años de edad, están madurando o han madurado sus habilidades cognitivas, las cuales se asemejan cada vez más a las de un adulto, sin embargo, mantienen intereses y gustos distintos. Debido a esto, el diseño de la interfaz se ha sustentado en los siguientes puntos clave: *Destreza física*, *Lectura*, *Estilo de interacción*, y *Conocimiento previo*.

6.4.1 Destreza física

El rendimiento de los niños con el ratón, el teclado y con otros dispositivos de entrada mejora con la edad. No obstante, pueden tener dificultades con tareas como: mantener presionado el botón del ratón por periodos prolongados, interactuar con elementos que requieren doble clic, o escribir textos extensos. Por ello, la interfaz del sistema está diseñada para no exigir del usuario más que un conocimiento y dominio

básico del teclado y del ratón. La figura 6.6 muestra la pantalla inicial del sistema, en ella se observa que el usuario debe emplear el ratón para interactuar con la herramienta.

Figura 6.6. Interacción del usuario con el sistema por medio del ratón.

6.4.2 Lectura

Los mensajes que despliega el sistema, son uno de los principales medios de comunicación entre el niño y la computadora. Consecuentemente, el léxico se ha elegido de modo que sea apropiado para el nivel de lectura de los niños. La letra tiene el color y tamaño adecuado para facilitar la lectura de los mensajes, tal como se muestra en la figura 6.7.

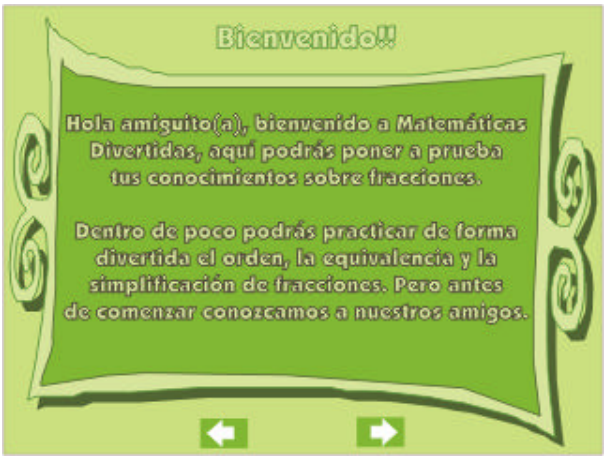

Figura 6.7. Mensaje del sistema.

6.4.3 Estilo de interacción

Los patrones de atención e interacción de los niños son un aspecto clave en el diseño de la interfaz. Si la interfaz tienen un gran número de animaciones, podría distraer a los usuarios durante el empleo de la herramienta, por el contrario, si es muy simple, podría no generar interés en ellos. Es menester mediar estos dos aspectos, de modo que se ofrezca una interfaz llamativa, y funcional. Por lo tanto, la herramienta hace uso de imágenes atractivas para el niño, dichas imágenes únicamente son animadas en casos necesarios, tales como indicar al usuario que la imagen es un botón. La figura 6.8, ilustra algunos botones de la herramienta.

Figura 6 .8. Botones de la interfaz.

6.4.4 Conocimiento previo

Es recomendable que la interfaz esté basada en metáforas familiares al niño. Las metáforas que se han empleado para la interfaz, son las de un mercado mexicano. Para lograr plasmar este escenario, se han empleado métodos de usabilidad tales como la metodología etnográfica [21]. Adicionalmente, para lograr que la herramienta sea más interesante, usable, y deseable para los niños, se ha solicitado el apoyo de estos para que brinden información que fortalezca el mapeo de la interfaz [15].

La metodología etnográfica, ha sido empleada para recopilar información sobre los detalles físicos que caracterizan a un mercado, tales como la disposición de los puestos, la localización de las personas y mercancías. Para llevar a efecto tal recopilación, se dio a la tarea de visitar mercados y típicas plazas comerciales que al ser fotografiadas facilitaran la representación en la computadora. En la figura 6.9, se muestra el menú principal de la aplicación, en él se observa la representación que se ha hecho de un mercado.

Figura 6.9. Menú del sistema, representación de un mercado mexicano.

Por otro lado, los niños han brindado información de los colores que les gustaría utilizar en la interfaz, así como los rasgos y nombres de los personajes (figura 6.10) que les agradaría que aparecieran en las actividades.

Lola Bongo Pedro

Figura 6.10. Personajes de la herramienta.

6.5 Resultado del modelado y mapeo de la interfaz

El modelado de la interfaz de las áreas de trabajo reconocidas a través del Modelo de Contenido, permitió crear una herramienta funcional, mientras que el mapeo de este modelo, definió los elementos de interacción. Cada mapeo siguió un desarrollo similar. Por ejemplo, en la figura 6.11, se muestra el mapeo del prototipo abstracto “Eligiendo un grupo de actividades”, y en la figura 6.12 se ilustra el mapeo del prototipo “Trabajando con la actividad. En ambas figuras, se observa la forma en que se han mapeado los materiales y las herramientas de cada prototipo.

Figura 6.11. Mapeo del área de trabajo “Eligiendo un grupo de actividades”.

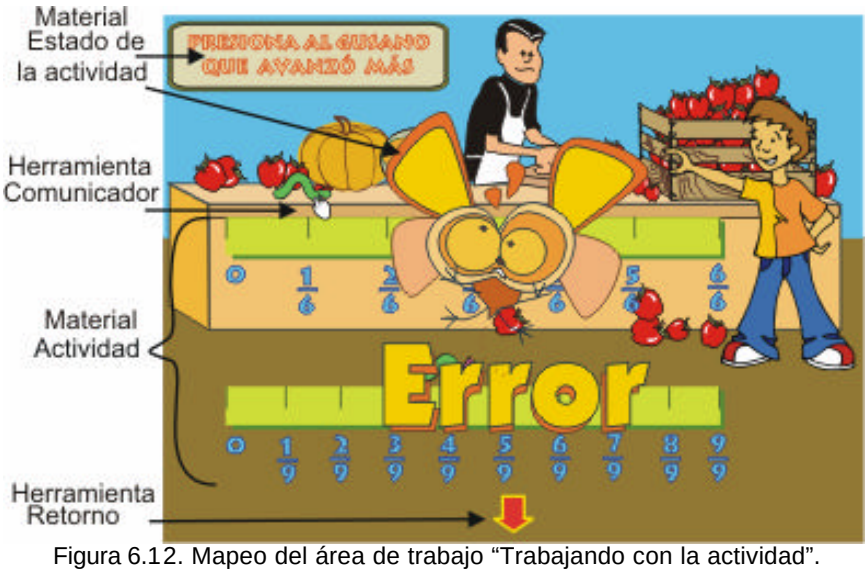

Figura 6.12. Mapeo del área de trabajo “Trabajando con la actividad”.

Los resultados del modelado y del mapeo, facilitaron y agilizaron el desarrollo de la interfaz de la herramienta, la cual se implementó con las bibliotecas de QT Designer en su versión de software libre. La interfaz resultante es funcional, atractiva, y permite que los usuarios practiquen las fracciones y las vinculen con actividades concretas.

Capítulo 7

Pruebas, mejoras y resultados

Este capítulo tiene como propósito exponer el trabajo realizado para cumplir con el último objetivo específico. En él, se habla de las pruebas de usabilidad realizadas al sistema, y se muestran las mejoras hechas conforme a las sugerencias generadas en estas pruebas. Finalmente presenta distintas pantallas de la herramienta que se obtuvieron al término de este trabajo.

7.1 Desarrollo de las pruebas de usabilidad

El principal objetivo de realizar las pruebas de usabilidad fue identificar y rectificar deficiencias de usabilidad en la interfaz de la herramienta. Además, se buscó corroborar el buen desempeño del sistema en las actividades. Para tal fin, se respetó el siguiente esquema de trabajo:

1. Planteamiento de las tareas de las pruebas, y del *test* de usabilidad.
2. Elección de los usuarios finales.
3. Representación del ambiente de trabajo real.
4. Observación de los usuarios finales durante el uso de la herramienta.
5. Recomendaciones de mejoramiento del diseño del producto.

Para el logro exitoso de las pruebas, fue necesaria la participación de distintas personas que desempeñaron roles de observador, usuario y de facilitador. El objetivo de cada uno de estos roles se menciona a continuación:

1. Rol de observador: Su objetivo fue escuchar, observar al usuario y al facilitador para tomar anotaciones relevantes durante las pruebas.
2. Rol de usuario: Su función fue la de probar el sistema y dar a conocer sus comentarios en relación a éste.
3. Rol de facilitador: Su misión fue la de guiar al usuario en el uso del sistema, así como hacer anotaciones de los comentarios expresados por los usuarios.

7.1.1 Planteamiento de las tareas de las pruebas y del *test* de usabilidad

Las pruebas de usabilidad comprendieron la realización de tres tareas, las cuales, tuvieron como interés evaluar el *Estilo de interacción*, la *Lectura*, y el *Apoyo que brinda*

la herramienta al proceso de aprendizaje de las fracciones. Enseguida se describen estas tareas.

Tarea 1. Estilo de interacción. Durante el diseño de los menús de la interfaz, se planteó integrar dos tipos distintos de botones. Los primeros botones se animaban únicamente cuando el puntero del ratón estaba sobre ellos, los segundos, se animaban todo el tiempo. Esta tarea tuvo como objetivo reconocer que tipo de botón prefería el usuario, así como la facilidad con la que los identificaban. Para tal fin, en esta tarea se planeó presentar al usuario dos menús del sistema, cada uno configurado con un único tipo de botones.

Tarea 2. Lectura. Aquí se propuso que el usuario trabajara con alguna actividad del sistema elegida por él. Al observar la forma en que el usuario empleaba la actividad, se analizaría que tan comprensibles y legibles eran para el usuario, los mensajes que brindaba el sistema antes y durante la actividad.

Tarea 3. Apoyo que brinda la herramienta. Con la finalidad de determinar si el sistema fortalecía el conocimiento del usuario, en esta tarea, se planteó que el usuario practicara el orden de fracciones con las actividades del sistema. Finalizada la práctica, se propuso cuestionar al usuario en cuanto a la manera en que la herramienta ejercita las fracciones en comparación con la forma tradicional de enseñanza.
Durante esta tarea, también se buscó evaluar el desempeño del sistema. Para lo que se propuso observar que éste proporcionará información correcta que permitiera al usuario la práctica adecuada de las actividades.

También, y con base a las tareas planteadas, se diseñó un *test* de usabilidad que tuvo como objetivo recabar los comentarios de los usuarios en cada tarea, así como saber si estos se sentían cómodos al utilizar el sistema.

7.1.2 Elección de los usuarios finales

Puesto que la herramienta esta dirigida a niños de sexto grado de primaria, se buscó que los usuarios que participaran en las pruebas estuvieran cursando o que hubieran cursado recientemente este grado escolar. Por tal motivo, y conforme a las sugerencias descritas en [15], las pruebas se realizaron con cinco niños que cumplían

este requisito. Además, y dado que es probable que los docentes hagan uso del sistema, la herramienta también fue probada por una pedagoga de educación básica. La figura 7.1 presenta a la docente y a los niños participantes.

7.1.3 Representación del ambiente de trabajo real

Los usuarios que participaron en las pruebas, utilizaron el sistema en un contexto similar al de una escuela que contara con la herramienta. La creación de este ambiente fue posible gracias al apoyo de la maestra de educación primaria, quien impartió una clase de fracciones a los niños. Posterior a la clase, la profesora y los niños usaron el software conforme a las tareas planeadas.

Figura 7.1. Docente y niños participantes.

7.1.4 Observación de los usuarios finales durante el uso de la herramienta

Durante la sesión de pruebas, el desempeño del sistema, las expresiones faciales, y la forma en que los usuarios interactuaron con la herramienta fueron observados y grabados para futuras revisiones. Las observaciones corrieron a cargo de profesores y técnicos de la Universidad Tecnológica de la Mixteca (figura 7.2), quienes desempeñaron el papel de observadores. Además, después de la elaboración de cada tarea, el facilitador(tesista) interrogó a los participantes sobre aspectos relevantes de las mismas (figura 7.3) conforme al *test* de usabilidad que se planteó.

Figura 7.2. Observadores de las pruebas.

Figura 7.3. Facilitador y niños durante una sesión de pruebas.

Algunos de los comentarios realizados por la docente y los niños durantes las sesiones de pruebas fueron:

Docente.

1. ¿Te agradó practicar el orden de fracciones en la computadora?
   Sí.
2. ¿Qué fue lo que más te agradó?
   Me agradaron las tres actividades, fue significativo partir de lo simple a lo complejo.
3. ¿Qué te agrada más, practicar en la computadora o en el pizarrón?
   En la computadora.
4. ¿Crees que utilizar el sistema, te facilite comprender el orden de fracciones?
   Sí, si ya tienes la noción, practicarlo o reforzarlo con el sistema le es más significativo al niño, se le queda más el conocimiento.

Niño 1.

1. ¿Te agradó practicar el orden de fracciones en la computadora?
   Sí.
2. ¿Qué fue lo que más te agradó?
   Me agradó las tres tareas, por que era de hacer cuentas mentalmente.
3. ¿Qué te agrada más, practicar en la computadora o en el pizarrón?
   En la computadora.
4. ¿Crees que utilizar el sistema, te facilite comprender el orden de fracciones?
   Sí.

Niño 2.

1. ¿Te agradó practicar el orden de fracciones en la computadora?
   Sí.
2. ¿Qué fue lo que más te agradó?
   Aprender con la computadora y al mismo tiempo jugar.
3. ¿Qué te agrada más, practicar en la computadora o en el pizarrón?
   En la computadora.
4. ¿Crees que utilizar el sistema, te facilite comprender el orden de fracciones?
   Sí, tiene dibujos para que se haga más fácil, y aparece si está bien o mal. En el salón la maestra necesita dibujarlo.

Niño 3.

1. ¿Te agradó practicar el orden de fracciones en la computadora?
   Sí.
2. ¿Qué fue lo que más te agradó?
   Hay muchos ejemplos de fracciones con frutas y verduras.
3. ¿Qué te agrada más, practicar en la computadora o en el pizarrón?
   En la computadora, solo hay que dar clic, no escribir.
4. ¿Crees que utilizar el sistema, te facilite comprender el orden de fracciones?
   Sí, se facilita más, lo explica más detalladamente.

Niño 4.

1. ¿Te agradó practicar el orden de fracciones en la computadora?
   Sí.
2. ¿Qué fue lo que más te agradó?

Los dibujos.

3. ¿Qué te agrada más, practicar en la computadora o en el pizarrón?
   En la computadora, por que te diviertes más.
4. ¿Crees que utilizar el sistema, te facilite comprender el orden de fracciones?
   Sí, por que te explica.

Niño 5.

1. ¿Te agradó practicar el orden de fracciones en la computadora?
   Sí, por que me gustaron los botones y las indicaciones.
2. ¿Qué fue lo que más te agradó?
   Practicar fracciones.
3. ¿Qué te agrada más, practicar en la computadora o en el pizarrón?
   En la computadora, por que es más llamativo, por que es más interesante por los monitos.
4. ¿Crees que utilizar el sistema, te facilite comprender el orden de fracciones?
   Sí, por los monitos y los colores pones más atención y facilita el aprendizaje.

7.1.5 Recomendaciones de mejoramiento del diseño del producto

Las anotaciones de los observadores, del facilitador, y las sugerencias de los usuarios, permitieron obtener una colección de recomendaciones de mejoramiento del diseño de la interfaz. Estas recomendaciones se listan a continuación:

Recomendaciones emergentes de la tarea 1:

1. Cambiar los botones que se animan únicamente cuando el puntero está sobre ellos, por botones completamente animados.
2. Agregar botones de continuar y regresar a los mensajes iniciales del sistema.

Recomendaciones emergentes de la tarea 2:

1. Agregar un mensaje que indique la función que se debe esperar del botón de regreso.
2. Reducir el texto de los mensajes iniciales del sistema.
3. Utilizar la misma combinación de colores en todos los mensajes de las distintas actividades que informan al niño que ha realizado bien o mal una tarea.

Recomendaciones emergentes de la tarea 3:

1. No se hicieron recomendaciones.

7.2 Mejoras realizadas

En base a las recomendaciones obtenidas durante la fase de pruebas, se modificó el sistema para su mejoramiento. Las figuras 7.4, 7.5, 7.6 y 7.7 muestran algunos de los cambios realizados al sistema.

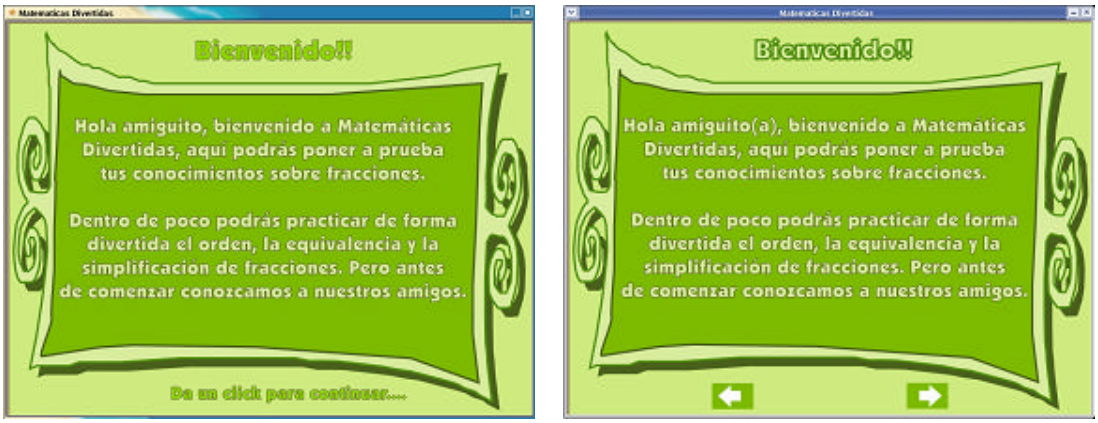

Figura 7.4. Mejora conforme a la recomendación 2 de la tarea 1.

Figura 7.5. Mejora conforme a la recomendación 1 de la tarea 2.

Figura 7.6. Mejora conforme a la recomendación 2 de la tarea 2.

Figura 7.7. Mejora conforme a la recomendación 3 de la tarea 2.

7.3 Herramienta resultante

Las pruebas realizadas al sistema han permitido mejorarlo en interacción y desempeño para la comodidad del usuario. Estos resultados incrementan las expectativas del proyecto al mostrar que el 100% de los niños y la docente confirman que la herramienta realmente es un apoyo a los procesos de aprendizaje. Las pantallas de las actividades del sistema, se muestran a continuación.

La figura 7.8, muestra la actividad de orden de nivel básico, esta actividad introduce el orden de fracciones en la recta numérica de forma gráfica.

Figura 7.8. Actividad de orden nivel básico.

La figura 7.9, presenta la actividad de orden de nivel intermedio, esta actividad combina las representaciones gráficas y textuales para ejercitar el orden de fracciones.

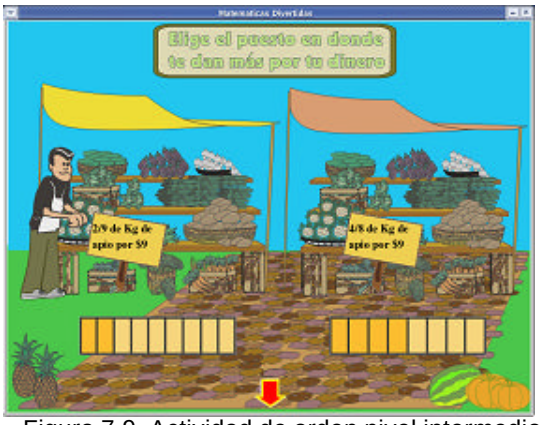

Figura 7.9. Actividad de orden nivel intermedio.

La actividad de orden de nivel avanzado, ilustrada en la figura 7.10, ejercita el orden de fracciones de forma textual, es por ello que en esta actividad, el niño debe hacer un mayor uso de su habilidad mental.

Figura 7.10. Actividad de orden nivel avanzado.

En la figura 7.11, se muestra la actividad de equivalencia de nivel intermedio. Esta actividad, ejercita el conocimiento del niño mediante representaciones textuales auxiliadas de breves consejos.

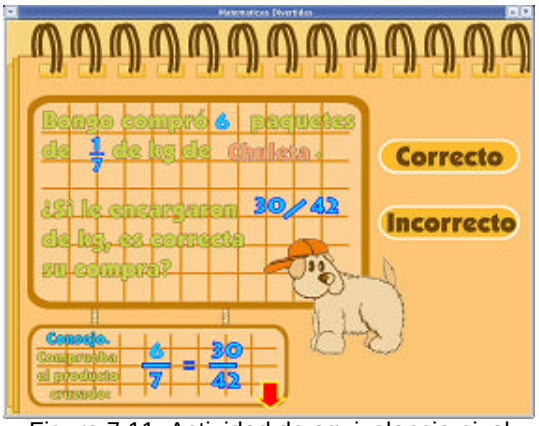

Figura 7.11. Actividad de equivalencia nivel intermedio.

La actividad mostrada en la figura 7.12, corresponde a la de equivalencia de nivel avanzado. Esta actividad, hace de uso de representaciones textuales para que el niño ejercite su conocimiento.

Figura 7.12. Actividad de equivalencia nivel avanzado.

La actividad de simplificación de fracciones de nivel avanzado, mostrada en la figura 7.13, ejercita los cálculos mentales del niño fomentando que haga uso del método numérico para la solución de los problemas.

Figura 7.13. Actividad de simplificación nivel avanzado.

Resultados y trabajos futuros

El esfuerzo realizado para madurar el proyecto, ha dado como resultado el logro exitoso de todos los objetivos, generándose una herramienta cuyas características:

- Fortalecen el aprendizaje de los alumnos en los subtemas *Equivalencia y orden entre fracciones* y *Simplificación de fracciones* del plan de estudio de las matemáticas de sexto grado de primaria propuesto por la SEP, debido al contexto e imágenes que emplean las actividades del sistema.
- Permiten que el alumno trabaje a un ritmo propio.
- Incrementan el interés del alumno por aprender.
- Explotan el potencial de la informática presentando escenarios que facilitan el aprendizaje.
- Vinculan el conocimiento abstracto con la realidad.
- Están acordes a las habilidades físicas y cognitivas del alumno.
- Están implementadas con software libre y gratuito, facilitando que las instituciones educativas e incluso los alumnos, puedan utilizarla sin implicar costos elevados.
- Permiten que los componentes didácticos empleados en el desarrollo de la herramienta, sean reutilizados para la creación de nuevo software.
- Facilita la integración de nuevos componentes didácticos en cuyo diseño intervengan pedagogos, diseñadores o personas afines.

Gracias a la obtención de una herramienta de software educativo de calidad, se ha podido participar en diversos foros de difusión mostrando las etapas de desarrollo del proyecto, logrando obtener retroalimentación que garantiza las mejoras en el proceso de desarrollo. Los foros donde se ha participado son los siguientes:

- Encuentro Internacional de Educación Superior UNAM. Virtual Educa 2005, Junio 2005. En este foro se presentó la arquitectura de distribución del sistema.
- Encuentro Internacional de Ciencias de la Computación. ENC 2005, Septiembre 2005. En este encuentro, se mostró en su totalidad el desarrollo que siguió la herramienta.

- Conferencia Latinoamericana de Interacción Humano-Computadora. CLHIC 2005, Octubre 2005. En esta conferencia se obtuvo retroalimentación acerca del proceso de diseño de la interfaz de la herramienta.

Los trabajos futuros de este proyecto consisten en:

- Realizar pruebas del apoyo que brinda la herramienta al aprendizaje de las fracciones en alguna institución primaria.
- Corroborar el funcionamiento de la herramienta en otros sistemas operativos.
- Analizar los problemas de enseñanza-aprendizaje de las matemáticas en el resto de los grados de educación primaria, y construir los componentes necesarios para anexarlos al sistema.

Apéndice A

Aplicaciones didácticas

Los distintos recursos de software que fueron estudiados en la fase de recopilación de requisitos de la herramienta didáctica gestada, se muestran en la tabla A.1. El contenido de la tabla se encuentra estructurado de la siguiente manera:

- *Nombre*. Nombre de la herramienta de software o recurso Web.
- *Descripción*. Descripción del material educativo sobre el que trabaja la aplicación, así como la forma en que interactúa con el usuario. Además, sugiere la etapa de aprendizaje más apropiada para el uso de la herramienta.
- *Subtemas.* Ejes que la aplicación cubre conforme a los planes y programas de estudio de las SEP.
- Características. Indica si el sistema trabaja de manera centralizada o en red, la plataforma en que corre, y si es un recurso libre o propietario.

Tabla A.1. Software didáctico para el apoyo al aprendizaje de las fracciones.

Nombre	**Descripción de la aplicación**	**Subtemas**	**Características**
Afracts12	Esta herramienta trabaja sobre las cuatro operaciones básicas de las fracciones. La aplicación genera un problema que debe resolver el usuario. La herramienta únicamente hace uso de información textual, por tal motivo, puede emplearse cuando el alumno comprende el concepto de fracción y está en condiciones de realizar cálculos mentales simples con estas [URL 14].	*Algoritmo convencional de la suma y resta de fracciones con igual denominador*, del cuarto grado de primaria.	Centralizada. Windows. Freeware.

Nombre	Descripción de la aplicación	Subtemas	Características
Calcul	Esta herramienta genera operaciones básicas sobre un par de fracciones propias o impropias con diferente denominador, el usuario debe dar respuesta al problema mediante una fracción mixta. La aplicación representa la información textualmente, por lo tanto, puede emplearse cuando el alumno comprende el concepto de fracción, y puede realizar cálculos mentales complejos con ellas [URL 5].	*Resolución de problemas de suma y resta de fracciones, y su representación en fracciones mixtas*, del cuarto y quinto grado de educación básica.	Centralizada. Windows. Freeware.
CalculPro	Es una aplicación que goza de las mismas características de Calcul, adicionalmente, funciona en ambiente de red, permitiendo el monitoreo de las actividades, así como el avance de los usuarios del sistema [URL 5].	*Resolución de problemas de suma y resta de fracciones, y su representación en fracciones mixtas*, del cuarto y quinto grado de educación básica.	Red. Windows. Shareware.
Clic	Es un ambiente para ejecutar aplicaciones interactivas agrupadas en paquetes de software. Dispone de una gran variedad de paquetes que fortalecen el conocimiento de las fracciones. Dentro de los paquetes disponibles, se encuentran: DivFrace, Espacri0, Fracc, Fraccio1, Fraccio2, Interface y Matsexto. Los contenidos de estos paquetes están dirigidos a niños con distintos niveles de conocimiento sobre las fracciones. En función de la complejidad de las actividades de los paquetes, la información se presenta en forma textual y/o gráfica [URL 7].	*Fraccionamiento de longitudes*, del quinto y sexto grado de primaria. *Suma y resta de fracciones con distinto denominador mediante el cálculo del denominador común*, de sexto grado. *Introducción a la noción de fracciones*, del tercer grado. *Equivalencia y orden de fracciones*, del sexto grado.	Centralizada. Windows, la versión Jclic funciona en Linux, Mac Os X, Solaris y Windows. Freeware.
CONEVyT .org	Es un portal mexicano de educación respaldado por diversas instituciones que buscan combatir el rezago educativo de los adultos. Debido a la forma gráfica y sonora con que el sistema interactúa con los usuarios, estos recursos pueden ser provechosos para fortalecer el aprendizaje de niños de primaria en etapas tempranas e intermedias [URL 2].	*Ubicación de fracciones en la recta numérica*, de sexto grado. *Equivalencia y orden de fracciones*, de sexto grado. Además de otros subtemas de sexto grado.	Página Web de acceso libre. Freeware.

Nombre	Descripción de la aplicación	Subtemas	Características
Escolar.com.	Esta página argentina resguarda una gran variedad de contenidos educativos gratuitos sobre distintas áreas de conocimiento de nivel primaria. Los contenidos son de gran utilidad en las etapas introductorias correspondientes, pues la información se presenta de manera textual, gráfica y sonora [URL 3].	*Suma y resta de fracciones con denominadores iguales*, de quinto grado. *Suma y resta de fracciones con denominadores distintos*, de sexto grado. *Equivalencia y simplificación de fracciones*, de sexto grado.	Página Web de acceso libre. Freeware.
Fract	Fract es una calculadora capaz de realizar operaciones básicas sobre un par de fracciones. Permite la conversión de fracciones a su equivalente decimal y viceversa. Es una aplicación útil para corroborar los resultados de las operaciones con fracciones. Representa la información de manera textual [URL 1].	No aplica dado que funciona como una calculadora simple.	Centralizada. Windows. Shareware.
Kbruch	Esta aplicación incorpora cuatro tipos de ejercicios: cálculo, comparación, conversión y factorización de fracciones. Kbrush es parte del KDE Edutainment Project, el cual busca crear aplicaciones educativas entretenidas [URL 6]. Kbrush es una aplicación sencilla que utiliza información textual para representar sus contenidos.	*Equivalencia y orden de fracciones*, de sexto grado. *Suma y resta de fracciones con distinto denominador mediante el cálculo del denominador común*, del sexto grado de primaria.	Centralizada. Libre. Linux.

Apéndice B

Configuración del cliente y del servidor

Configuración de las variables de ambiente del cliente y del servidor

Para que el cliente y el servidor funcionen apropiadamente, es necesario que se configuren las variables de ambiente y se establezcan las rutas en las que residen las bibliotecas de MICO. Esta configuración se realiza en el archivo /etc/profile de Linux. Enseguida, se presenta las configuraciones que son necesarias agregar a este archivo.

```
#Configuración para MICO

prefix="ruta_en_la_que_se_ha_instalado_MICO"
exec_prefix="${prefix}"
MICODIR="$exec_prefix"
MICOSHAREDDIR="$prefix"
MICOVERSION=` sed -n '/MICO_VERSION/ { y/b/./;
s#^[^"]*"\([^"]*\)".*$#\1#p; }' \
  "$MICODIR/include/mico/version.h" `
PATH="$MICODIR/bin:$PATH"
LD_LIBRARY_PATH="$MICODIR/lib:${LD_LIBRARY_PATH:-}"
SHLIB_PATH="$MICODIR/lib:${SHLIB_PATH:-}"
LIBPATH="$MICODIR/lib:${LIBPATH:-}"
MANPATH="$MICOSHAREDDIR/man:${MANPATH:-}"
CPLUS_INCLUDE_PATH="$MICODIR/include"
LIBRARY_PATH="$MICODIR/lib"

export MICOVERSION PATH LD_LIBRARY_PATH MANPATH
CPLUS_INCLUDE_PATH
LIBRARY_PATH
export SHLIB_PATH LIBPATH MICODIR

unset prefix
unset exec_prefix
```

Configuración para que el servidor sea reconocido en la red

Las máquinas que utilizan el sistema operativo Linux, generalmente están configuradas para reportar su nombre en la red como *localhost*. Sin embargo, para que

los clientes puedan conectarse con el servidor, se requiere que el sistema operativo obtenga el nombre de la máquina y lo relacione con la dirección IP correspondiente. Para solucionar este problema de configuración, es necesario modificar la variable HOSTNAME en el archivo /etc/profile de este modo:

```
#Configuración para ambiente de red

hostname `ifconfig eth0 | grep inet | sed 's/.*r:\|  B.*//g'`
```

Apéndice C

Código fuente de las clases y operaciones básicas

Código de la definición de interfaces del archivo fracciones.idl

```
module Fracciones_Sexto{

  //Declaracion del tipo de dato fracción
  struct Fraccion{
                long numerador;
                long denominador;
                boolean positivo;
               };

  //Interfaz que tiene operaciones necesarias para crear fracciones y modificar sus datos
  interface Crea_Fraccion{
                //Crea una fracción inicializada con los valores contenidos en los parámetros
                 Fraccion nueva_fraccion(in long num,in long den,in boolean pos) ;

                 //Crea una fracción positiva inicializada con valores aleatorios
                 //El numerador se genera en el rango   [nli-nls]
                //El denominador se genera en el rango [dli-dls]
                 Fraccion nueva_fraccion_positiva (in long nli, in long nls, in long dli, in long dls);

                 //Crea una fraccion negativa inicializada con valores aleatorios
                 //El numerador se genera en el rango   [nli-nls]
                //El denominador se genera en el rango [dli-dls]
                 Fraccion nueva_fraccion_negativa (in long nli, in long nls, in long dli, in long dls);

                 //Copia una fracción
                 Fraccion copia_fraccion (in Fraccion f);
                };

  //Interfaz que contiene los métodos necesarios para obtener la relación de orden entre
fracciones
  interface Orden : Crea_Fraccion  {
                //Obtiene la relacion de orden entre un par de fracciones
                 //Esta funcion regresa '-1' si f1 < f2, '0' si f1 = f2 o '1' si f1 > f2
                 long relacion_orden(in Fraccion f1, in Fraccion f2);
                };

  //Interfaz cuyos métodos permiten determinar si las fracciones dadas son equivalentes
  interface Equivalencia : Crea_Fraccion  {
                 //Analiza si dos fracciones son exactamente iguales
                 boolean son_iguales     (in Fraccion f1, in Fraccion f2);

                //Analiza si dos fracciones son equivalentes
                 boolean son_equivalentes(in Fraccion f1, in Fraccion f2);
```

```
            //Retorna una fraccion equivalente a la proporcionada como parametro
             Fraccion obten_equivalente (in Fraccion f);
            };

 //Interfaz que proporciona funcionalidad para simplificar fracciones
 interface Simplificacion : Equivalencia {
            //Obtiene una fraccion simplificada de la proporcionada como parametro
             Fraccion simplifica_fraccion (in Fraccion f);

            };
};
```

Código de las principales operaciones de la clase Control_actividad del cliente

```
//Establece una referencia al objeto remoto
void Control_actividad::set_reference(const char* nombre_objeto)
{
 try {
      extern CORBA::ORB_var global_orb;

       //1. Obtengo una referencia al servicio de nombres CORBA
       CORBA::Object_var objV = global_orb-> resolve_initial_references ("NameService");

       //2. Los nombres de los objetos dados de alta en el servicio de nombres, están
       //    organizados en un árbol n -ario. Con esta instrucción obtengo una referencia a la raíz
       //    del árbol de nombres
       rootContextExtV = CosNaming::NamingContextExt::_narrow( objV.in() );

       //3. Obtengo la referencia al servidor
       obj_ref = rootContextExtV->resolve_str(nombre_objeto);
      }
 catch (CORBA::SystemException &sysEx) {
       cerr << sysEx << endl;
    }
}


//Invoca una funcion en el objeto referenciado
void Control_actividad::invoca (const char *funcion, struct Argumentos &arg)
{
 //1. Obtengo el nombre de la interfaz del objeto remoto
 CORBA::InterfaceDef_var if_def = obj_ref->_get_interface();

 //2. Obtengo la descripción completa de la interfaz
 CORBA::InterfaceDef::FullInterfaceDescription_var full_if_desc = if_def -> describe_interface
();

 //2.1 Determ ino si existe la operación deseada en la interfaz del objeto
 unsigned int funcion_id;
 for (funcion_id=0; funcion_id < full_if_desc->operations.length(); funcion_id++)
     if( strcmp(full_if_desc->operations[funcion_id].name, funcion) == 0 )
         break;
 if( funcion_id == full_if_desc->operations.length() ){
     cout << "La funcion " << funcion << " no pertenece al objeto" << endl;
```

```
    exit(1);
  }

 //3. Creo e inicializo el valor de retorno
 CORBA::NVList_ptr result_list = new CORBA::NVList();
 result_list->add_value("resultado", CORBA::Any(full_if_desc->operations[funcion_id].result,0),
0 );

 //3.1 Creo e inicializo la lista de argumentos
 CORBA::NVList_ptr arg_list = new CORBA::NVList();
 arg_list = crea_lista_argumentos(full_if_desc, funcion_id, arg);

//4. Creo la solicitud
 CORBA::Request_var request;
 obj_ref->_create_request(
                CORBA::Context::_nil(),                    //Contexto – no usado
                full_if_desc->operations[funcion_id].name, //Nombre de la operación
                arg_list,                                  //Lista de argumentos
                result_list->item(0),                      //Valor de retorno
                request.out(),                             //Salida – objeto solicitado creado
                0                                          //Banderas
               );

 //5. Invocación de la solicitud
 try {
     request->invoke();
    }
 catch ( CORBA::TRANSIENT &e){
     cerr << "Error al intentar conectarse al servidor: " << e << endl;
    }
 catch ( CORBA::COMM_FAILURE &e){
     cerr << "La conexión al servidor se rompió: " << e << endl;
    }
 catch ( CORBA::OBJECT_NOT_EXIST &e){
     cerr << "El objeto buscado no se encuentra: " << e << endl;
    }
 catch ( CORBA::BAD_INV_ORDER &e){
     cerr << "La operación fue invocada con un orden erróneo: " << e << endl;
    }
 catch ( CORBA::BAD_OPERATION &e){
     cerr << "La operación no existe en el objeto servidor, al parecer el objeto servidor no existe:
" << e << endl;
    }
 catch ( CORBA::BAD_PARAM &e){
     cerr << "Un parámetro es de algún tipo distinto al requerido, o su valor es inaceptable: " <<
e << endl;
    }
 catch ( CORBA::MARSHAL &e){
     cerr << "La petición al GIOP o el mensaje de retorno esta malformado: " << e << endl;
    }

 catch ( CORBA::UNKNOWN &e){
     cerr << "Se desconoce la causa de error: " << e << endl;
    }

 //6. Extrayendo el valor de retorno
 CORBA::Any *res_any_var = request->result()->value();
```

```
//Checando el tipo del valor de retorno
CORBA::TypeCode_var tc = res_any_var->type();

extern Fracciones_Sexto::Fraccion global_fraccion;
extern CORBA::Long                global_long;
extern CORBA::Boolean             global_boolean;

if ( tc->kind() != CORBA::tk_any ){
     //El valor de retorno no es any
     if( !(*res_any_var >>= global_fraccion) )
       if( !(*res_any_var >>= CORBA::Any::to_boolean(global_boolean)) )
           if( !(*res_any_var >>= global_long) ) {
               cout << "Warning: Tipo de retorno inesperado" << endl;
            }
    }
 else{
     cout << "Warning: Tipo de retorno inesperado" << endl;
     cout << "Warning: Retorno es un any, será necesario extraerlo"<<  endl ;
    }
}
```

Código de la aplicación servidor y de las principales operaciones de la clase Crea_Fraccion_impl

Aplicación servidor

```
int main (int argc, char *argv[])
{
  //Declaro las variables para almacenar las referencias al objeto servidor
  PortableServer::ServantBase_var servidor = 0;

  srand ( (unsigned) time (0));
  try {
       //Auxiliar para almacenar referencias temporales
       CORBA::Object_var objV;

       //Paso 1. Inicializando el ORB
       global_orb = CORBA::ORB_init (argc, argv);
       if (CORBA::is_nil(global_orb.in())){
           cerr << "Referencia vacia al objeto ORB" << endl;
           return 1;
          }

      //Paso 2. Obteniendo una referencia al RootPO A
       objV = global_orb->resolve_initial_references ("RootPOA");
       PortableServer::POA_var root_poaV = PortableServer::POA::_narrow (objV);
       if (CORBA::is_nil(root_poaV.in())){
           cerr << "Error al hacer narrow al root POA" << endl;
           return 1;
          }

      //Obteniendo una referencia al POA Manager
       PortableServer::POAManager_var root_poa_managerV = root_poaV->the_POAManager();
       if (CORBA::is_nil(root_poa_managerV.in())){
           cerr << "Error al hacer narrow al root POA manager" << endl;
```

```
        return 1;
       }

    //Paso 3. Obteniendo una referencia al servio de nombres CosNaming::NamingContextExt
     CosNaming::NamingContextExt_var rootContextExtV; // INS Root ContextExt
     try {
         objV = global_orb -> resolve_initial_references ("NameService");
         rootContextExtV = CosNaming::NamingContextExt::_narrow(objV.in() );
        }
     catch (CORBA::SystemException &sysEx) {
           cerr << sysEx << endl;
         return 1;
        }
     if (CORBA::is_nil(rootContextExtV.in() )) {
         cerr << "Nil root naming context" << endl;
         return 1;
        }

    //Paso 4. Crear y activar los servidores
     CosNaming::Name_var nameV;

     //Paso 4.1 Crea y activa servidor Crea_Fraccion_impl
    servidor = new Crea_Fraccion_impl();

     //Activando el objeto
    PortableServer::ObjectId_var oid = root_poaV->activate_object (servidor);
    CORBA::Object_var ref = root_poaV->id_to_reference (oid.in());

     try {
         nameV = rootContextExtV->to_name("FraccionesSexto.fracciones");

         //Creo el contexto al que pertenecerá el objeto
         createContextPath( rootContextExtV.in(), nameV.in() );

         //Creo la relación nombre-referencia
         nameV = rootContextExtV->to_name("FraccionesSexto.fracciones/Crea" );
         bindObjectPath( rootContextExtV.in(), nameV.in(), ref.in());
        }
     catch (CORBA::Sys temException& se) {
         cerr << se << endl;
        return 1;
        }

    //Paso 5. Activando el POA manager y esperando peticiones
     root_poa_managerV -> activate();

     //Permitiendo que el ORB procese las peticiones
     cout << "Esperando peticiones ..." << endl;
     global_orb ->run();
    }


catch (CORBA::Exception& e) {
     cout << "Excepción CORBA: " << e << endl;
    }

try {
    //Destruyendo el ORB
```

```
      global_orb -> destroy();

      //El servidor se destruye automáticamente
     }

 catch (...){
      //No hace nada
     }

 return 0;
}
```

Clase Crea_Fraccion_impl

```
/* Función heredada de ServantBase
 * Esta función debe regresar un solo repositorio en forma de cadena (char *) ID
 * que identifi que la interfaz del objeto implementado */
CORBA::RepositoryId Crea_Fraccion_impl::_primary_interface
 (const PortableServer::ObjectId& oid, PortableServer::POA_ptr poa )
 {
  return CORBA::string_dup((const char *)"IDL:Fracciones_Sexto/Crea_Fraccion:1.0");
 }

/* El método invoke, heredado de DynamicImplementation, debe proporcionar la lógica para
procesar las invocaciones DSI. Cuando se recibe una solicitud a una función, se pasa la
petición en la forma de un objeto CORBA::ServerRequest_ptr al sirviente D SI mediante la
invocación de este método */
void Crea_Fraccion_impl::invoke(CORBA::ServerRequest_ptr request)
 {
  //1. Determino el nombre de la operación invocada y delego el procesamiento
  //    el procesamiento a la operación correspondiente
  if( strcmp( request->op_name(), "nueva_fraccion") == 0 )
      nueva_fraccion_impl(request);
  else if( strcmp( request->op_name(), "nueva_fraccion_positiva") == 0 )
          nueva_fraccion_positiva_impl(request);
  else if( strcmp( request->op_name(), "nueva_fraccion_negativa") == 0 )
          nueva_fraccion_negativa_impl(request);
  else throw CORBA::BAD_OPERATION();
 }


//Crea una fracción positiva inicializada con valores aleatorios
//el numerador   se genera con un valor entre [nli-nls]
//el denominador se genera con un valor entre [dli-dls]
void Crea_Fraccion_impl::nueva_fraccion_positiva_impl  (CORBA::ServerRequest_ptr request)
{
 //2. Creo la lista en la que se recuperaran los argumentos
 CORBA::NVList_ptr params = new CORBA::NVList(4);

 //2.1 Agrego los argumentos con valor y tipo
 CORBA::Any any;

CORBA::Long l=0;
 any <<= l;
 params->add_value("nli", any, CORBA::ARG_IN);
```

```
 params->add_value("nls", any, CORBA::ARG_IN);
 params->add_value("dli", any, CORBA::ARG_IN);
 params->add_value("dls", any, CORBA::ARG_IN);

//3. Obtengo los argumentos y los extraigo
request->arguments(params);

 //3.1 Extraigo los argumentos de la lista
 CORBA::Long nli, nls, dli, dls;
 *(params->item(0)->value()) >>= nli;
 *(params->item(1)->value()) >>= nls;
 *(params->item(2)->value()) >>= dli;
*(params->item(3)->value()) >>= dls;

 //4. Proceso la solicitud
 int n = rand() % 44;

 Fracciones_Sexto::Fraccion f;
 f.numerador   = fracciones[0][n]/*nli+rand()%(nls-nli)*/;
 f.denominador = fracciones[1][n]/*dli+rand()%(dls-dli)*/;
 f.positivo = true;

 //5. Coloco el valor de retorno
 CORBA::Any result;
 result <<= f;
 request->set_result(result);
}
```

Apéndice D

Casos de uso y prototipos abstractos del modelado de la interfaz de la herramienta

D.1 Casos de uso

Los casos de uso que fueron identificados en el modelo de casos de uso, se muestran a continuación:

Identificación	
ID 01	**Nombre** eligiendo un grupo de actividades
Objetivo Contextual Para trabajar con actividades de un subtema, el usuario debe elegir un grupo de actividades	**Roles Soportados** Alumno, Profesor
Relaciones	
Especializa a	**Extiende a**
Parecido a eligiendo una actividad	**Equivale a**
Proceso	
Precondiciones Un usuario ha iniciado la aplicación	
Intenciones del Usuario *Extensión asíncrona* Opcionalmente y en cualquier punto, salir de la aplicación	**Respuesta del Sistema** *Extensión asíncrona* Opcionalmente y en cualquier punto, reportando una excepción fatal

2. Elegir un grupo de actividades	1. Presentar los grupos de actividades disponibles
Post Condiciones Se he elegido un grupo de actividades e iniciado eligiendo una actividad	

Identificación

ID	Nombre
02	eligiendo una actividad
Objetivo Contextual	**Roles Soportados**
Para trabajar con una actividad, el usuario debe elegirla dentro de un grupo de actividades	Alumno, Profesor

Relaciones

Especializa a	Extiende a
Parecido a	**Equivale a**
eligiendo un grupo de actividades	

Proceso

Precondiciones
Un usuario ha elegido un grupo de actividades con eligiendo un grupo de actividades

Intenciones del Usuario	Intenciones del Usuario
Extensión asíncrona	*Extensión asíncrona*
Opcionalmente y en cualquier punto, salir de la aplicación; regresar a eligiendo un grupo de actividades	Opcionalmente y en cualquier punto, reportando una excepción fatal
	1. Presentar las actividades del grupo elegido
2. Elegir una actividad	

Post Condiciones
Se ha elegido una actividad

Identificación

ID	Nombre
03	trabajando con la actividad
Objetivo Contextual	**Roles Soportados**
Dirigir o reforzar el aprendizaje de un subtema específico mediante actividades didácticas	Alumno, Profesor

Relaciones

Especializa a	Extiende a
Parecido a	**Equivale a**

Proceso

Precondiciones
Un usuario ha elegido trabajar con una actividad mediante eligiendo una actividad

Intenciones del Usuario	Respuesta del Sistema
Extensión asíncrona	*Extensión asíncrona*
Opcionalmente y en cualquier punto, salir de la aplicación; reiniciar la actividad; regresar a eligiendo una actividad	Opcionalmente y en cualquier punto, reportando una excepción fatal
	1. Presentar la actividad al usuario
2. Interactuar con la actividad	3. Retroalimentar al usuario
4. Finalizar la actividad	

Post Condiciones

Identificación	
ID 04	**Nombre** Reportando una excepción fatal
Objetivo Contextual Informar al usuario que ha ocurrido un error en la aplicación que impide el correcto funcionamiento de ésta	**Roles Soportados** Alumno, Profesor
Relaciones	
Especializa a	**Extiende a**
Parecido a	**Equivale a**
Proceso	
Precondiciones Se ha originado un error propio de la aplicación	
Intenciones del Usuario	**Respuesta del Sistema**
Extensión asíncrona	*Extensión asíncrona*
	1. Informar del problema y presentar las opciones que el sistema ofrece como solución
2. Elegir una opción	3. Ejecutar opción
Post Condiciones	

D.2 Prototipos abstractos del modelo de contenido

A continuación, se presentan los prototipos abstractos del modelo de contenido que se presentó en el apartado 6.3, respetando la simbología mostrada en la figura D.1 para las herramientas y materiales.

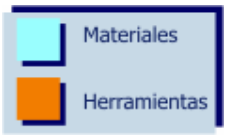

Figura D.1. Simbología utilizada para los modelos de contenidos.

Prototipo abstracto para el caso de uso "Eligiendo un grupo de actividades".

Tal como se ilustra en la figura D.2, éste prototipo consta de los siguientes elementos:

Materiales.

- Contenedor de grupos de actividades. Elemento de la interfaz que contiene los distintos grupos de actividades soportados por el sistema.
- Informador de grupo. Brinda información breve sobre cada grupo de actividades, permitiendo que el usuario elija con mayor certeza el grupo sobre el que desea trabajar.

Herramientas.

- Seleccionador de grupo de actividad. Herramienta que permite escoger un grupo de actividades del contenedor de grupos de actividades.
- Terminador de aplicación. Elemento que facilita al usuario finalizar la herramienta en un momento arbitrario.

Prototipo abstracto para el caso de uso "Eligiendo una actividad".

El prototipo mostrado en la figura D.3, se integra de los siguientes elementos:

Materiales.

- Contenedor de actividades. Elemento cuya función es la de contener las distintas actividades pertenecientes a un grupo determinado.
- Informador de actividad. Este material proporciona información breve sobre el objetivo particular de las actividades, facilitando la elección del usuario.

Herramientas.

- Seleccionador de actividad. Herramienta que permite seleccionar una actividad del contenedor de actividades.
- Retorno a eligiendo un grupo de actividades. Esta herramienta tiene como objetivo permitir que el usuario se desplace al menú anterior.

Figura D.2. Prototipo abstracto para el caso de uso esencial "Eligiendo un grupo de actividades".

Figura D.3. Prototipo abstracto para el caso de uso esencial "Eligiendo una actividad".

Prototipo abstracto para el caso de uso "Trabajando con la actividad".

La figura D.4 muestra este prototipo, los elementos de la interfaz que intervienen en él se desglosan a continuación:

Materiales.

- Actividad. Este material es el elemento pilar en la interfaz del usuario, pues con él, el alumno trabaja para reforzar su aprendizaje

- Estado de la actividad. Este elemento de la interfaz, retroalimenta al usuario sobre su desempeño actual con la actividad.

Herramientas.

- Retorno a eligiendo una actividad. Esta herramienta tiene como objetivo permitir que el usuario se desplace al menú anterior.
- Comunicador. Herramienta que permite que el usuario interactúe con el material actividad.

Figura D.4 . Prototipo abstracto para el caso de uso "Trabajando con la actividad".

*Prototipo abstracto para el caso de uso "*Reportando una excepción fatal*".*

Enseguida se analizan los componentes de esta área de trabajo, la cual está representada en la figura D.5

Materiales.

- Excepción. Este material informa al usuario de un error fatal del sistema, al mismo tiempo le brinda alternativas para restaurar el sistema.

Herramientas.

- Seleccionador de opción. Herramienta que facilita al usuario elegir una de las opciones presentadas al momento de que el sistema marque una excepción.

Figura D.5. Prototipo abstracto para el caso de uso esencial “Reportando una excepción fatal”.

Bibliografía

[1]. Amoretti, M. et al. 2003. Experience in Teleoperation System Design based on Real-Time CORBA. *Eleventh Internacional Conference on Advanced Robotics* (ICAR 11, Coimbra, Portugal, Junio).

[2]. Apezteguia, E. et al. 2000. CORBA vs. DCOM vs. RMI. Reporte técnico. Universidad de Navarra, España.

[3]. Bolton, F. 2002. *Pure CORBA*. Sams Publising, United Status of America.

[4]. Bruckman, A., Bandlow, A. 2002. *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications.* Lawrence Erlbaum and Associates.

[5]. C. George. 2001. *Distributed systems concepts and design*. Addison Wesley, Estados Unidos.

[6]. Caituiro, H. 2002. CORBA. Reporte técnico. Departamento de Ingeniería Eléctrica y de Computadoras, Facultad de Mayagüez, Universidad de Puerto Rico.

[7]. Cardona, G. 2002. Tendencias educativas para el siglo XXI. *Edutec, Revista Electrónica de Tecnología Educativa*, 15 (Dic.).

[8]. Casarrubias A. 1993. *Matemáticas didáctica 1* Ediciones Alegre Juventud, México.

[9]. Chapman, D., Mählck L. 2004. *Adapting technology for school improvement: a global perspective. Reporte técnico*. International Institute for Educational Planning, Paris.

[10]. Constantine, L. 1998. Rapid abstract prototyping. Software Development.

[11]. Constantine, L., Lockwood, L. 2001. *Object-Modeling and User Interface Design*. Addison-Wesley.

[12]. Dirección General de Materiales y Métodos Educativos de la Subsecretaría de Educación Básica y Normal. 1994. *Libro para el Maestro, Matemáticas Sexto Grado*. Secretaría de educación pública, México.

[13]. Dirección General de Materiales y Métodos Educativos de la Subsecretaría de Educación Básica y Normal. 1997. *Libro para el Alumno, Matemáticas Sexto Grado*. Secretaría de educación pública, México.

[14]. Dirección General de Materiales y Métodos Educativos de la Subsecretaría de Educación Básica y Normal. 2000. *Fichero de Actividades Didácticas, Matemáticas Sexto Grado*. Secretaría de educación pública, México.

[15]. Druin, A. 1999. The role of children in the design of new technology. Reporte técnico HCIL No 99-23. Universidad de Maryland.

[16]. Hyde, R. 2003. Mathsalive – developing a technologically-rich learning environment for secondary mathematics. *Third Conference of the European Society for Research in Mathematics Education* (CERME 3, Bellaria, Italia, Febrero).

[17]. OMG. 2002. UML Profile for CORBA Specification, OMG.

[18]. Orfali, R., Harhey, D. *Client/Server Programming with Java and CORBA*. Wiley Computer Publishing, 1998.

[19]. Parzysz, B., Bergsten, C., et al. 2003. Role of metaphors and images in learning and teaching mathematics. *Third Conference of the European Society for Research in Mathematics Education* (CERME 3, Bellaria, Italia, Febrero).

[20]. Perkkilä, P. 2003. Primary school teachers' mathematics beliefs and teaching practices. *Third Conference of the European Society for Research in Mathematics Education* (CERME 3, Bellaria, Italia, Febrero).

[21]. Rosenbaum, S. 2002. Usability in Practice: Field Studies. Field Studies – Evolution and Revolution CHI.

[22]. Secretaría de Educación Pública, Instituto Latinoamericano de la Comunicación Educativa. 1999. Enseñanza de las matemáticas y enseñanza de la física con tecnología (EMAT y EFIT). *Revista Red Escolar*, 1 (Abr.).

[23]. Secretaría de Educación Pública. 1993. *Plan y programas de estudio 1993*. Secretaría de educación pública, México.

[24]. Tamez, R., 2003. A mitad de la jornada, avances en la educación 2001-2003. Secretaría de educación pública, México.

[25]. Tamez, R., Fraustro, J., et. 2001. *Programa Nacional de Educación 2001-2006*. Secretaría de educación pública, México.

[26]. Tamez, R., Medellín, F., et. 2002. *Prácticas Educativas Innovadoras en las Entidades Federativas*. Secretaría de educación pública, México.

URL´s

[1]. Breaktru. Product info and donwload. Consultado el 05/11/2004 disponible en http://www.breaktru.com/.

[2]. Conevyt. Fracciones y porcentajes. Consultado el 05/11/2004 disponible en http://carolina.conevyt.org.mx/cursos/fracciones/curso.htm.

[3]. Escolar. Operaciones con fracciones. Consultado el 05/11/2004 disponible en http://www.escolar.com/matem/09opfrac.htm.

[4]. FreeWare World Team. Alan's Fractions. Consultado el 05/11/2004, disponible en http://www.all4you.dk/FreewareWorld/links.php?id=13936.

[5]. Luzius-Schneider. Luzius-Schneider Software Products. Consultado el 05/11/2004, disponible en http://www.filesland.com/companies/Luzius-Schneider/products.html.

[6]. The KDE Edutainment Project. The KDE Edutainment Project. Consultado el 05/11/2004 disponible en http://edu.kde.org/.

[7]. Zona Clic. Zona Clic. Consultado el 05/11/2004 disponible en http://clic.xtec.net/es/index.htm.