SAC: sistema informático para control de actividades complementarias. Módulo Alumnos

M.G. Flores-Luévanos[1], E. Moreno-Nuñez[1], E.V.J Solorio-Vega[2]

***Resumen*— La presente investigación se centrará en el análisis y diseño de la versión para los alumnos del Sistema de Actividades Complementarias (SAC).**

Se describe de manera general en qué consisten las actividades complementarias que se realizan en los Institutos Tecnológicos de nuestro país y en específico en el Instituto Tecnológico Superior de Lerdo, el proceso para llevarlas a cabo y acreditarlas.

El objetivo de esta investigación fue identificar las herramientas de diseño del sistema informático SAC y contextualizar su entorno de aplicación, específicamente desarrollar e implementar en el SAC un módulo que permita a los estudiantes consultar su estado en las actividades complementarias en las que están inscritos. La metodología de desarrollo de software fue orientada a objetos y se utilizó UML (Lenguaje Unificado de Modelado, del inglés *Unified Modeling Language*) como lenguaje de modelado. Se presentan los productos principales del análisis y diseño y pantallas del software. Al final se incluyen las conclusiones de este proceso.

***Temas claves*—Actividades complementarias, proceso, análisis, desarrollo, software.**

***Abstract* – This research will focus on the analysis and design of the "Complementary Activities System" (SAC) software, version for students.**

It generally describes the complementary activities that take place in the Technological Institutes of our country (México) and specifically in the Instituto Tecnológico Superior de Lerdo, the process to carry out and accredit .

The objective of this research was identify design tools SAC computer system and contextualize its application environment, specifically develop and implement in SAC a module that allows students check its progress of complementary activities which are enrolled. The software development methodology is OOP and UML is used as a modeling language. The main products of analysis and software design are presented and described the methodology used for development. Finally the conclusions of this process are included.

***Keywords* — Complementary activities, process, analysis, design, software.**

I. INTRODUCCIÓN

El desarrollo de software implica mucho más que escribir instrucciones de programación y ejecutarlas en una computadora. Se requiere cumplir los requisitos del cliente de acuerdo a una planificación preestablecida. Para tener éxito y obtener productos de calidad, los ingenieros de software deben regirse por un proceso de desarrollo de calidad.

Las actividades complementarias son todas aquellas que realiza el estudiante de los Institutos Tecnológicos, en beneficio de su formación integral con el objetivo de complementar sus competencias profesionales. Éstas pueden ser: tutorías, actividades extraescolares, proyectos de investigación, participación en eventos académicos, productividad laboral, emprendimiento, fomento a la lectura, construcción de prototipos y desarrollo tecnológico, conservación al medio ambiente y participación en ediciones, o aquellas que defina el comité académico.

El valor curricular para el conjunto de las actividades complementarias establecidas en el plan de estudios es de 5 créditos, considerando que por cada crédito equivale a 20 horas y su cumplimiento debe ser dentro de los seis primeros semestres.

Cada una de las actividades complementarias autorizadas por el plantel, no deben de tener más de 2 créditos. En el ITSL estas actividades tienen un valor curricular de un crédito cada una.

A. Fundamento Teórico

Para llevar a cabo este proyecto, se hará uso de las herramientas tecnológicas siguientes: Project, UML así mismo para el desarrollo se usarán CSS, ASP, SQL, Visual Studio: C#.

B. UML

UML permite una modelación de los componentes estáticos de una aplicación software (diagramas de casos de uso, diagramas de clases), así como del comportamiento dinámico de sus principales elementos durante su funcionamiento (entre ellos diagramas de estados y diagramas de secuencias). Los diagramas de estados permiten la modelación de los principales estados y los eventos que ocasionan sus cambios para una instancia de una clase, o para un sistema como un todo, mientras que los diagramas de secuencias permiten modelar instancias de interacción entre actores u objetos de clases de un sistema a través de mensajes. Mediante estos últimos diagramas es posible conocer lo que ocurre internamente entre los actores e instancias de clases que

participan en un diagrama de estados de un sistema de software. [1]

Entre sus principales características:
Soporte completo al diseño UML mediante el uso de:

- Diagrama de casos de uso
- Diagrama de clase
- Diagrama de secuencia
- Diagrama de colaboración
- Diagrama de estados
- Diagrama de actividad.

[2]

C. Microsoft Project
Project ayuda a planificar proyectos y a colaborar con otras personas fácilmente teniendo todo organizado y realizando el seguimiento de los proyectos con el único sistema de administración de proyectos diseñado para trabajar sin ningún problema con otras aplicaciones de Microsoft y servicios en la nube. [3]

D. *Metodología***.**
La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto con altas posibilidades de éxito, esta sistematización indica cómo se divide un proyecto en módulos pequeños llamados etapas y las acciones que corresponden a cada una de ellas, ayuda a definir entradas y salidas para cada una y normaliza el modo en que se administrará el proyecto.
Existen dos metodologías que tienen analogía en la práctica con los paradigmas de programación: metodología estructurada y metodología orientada a objetos.

II. PARTE TÉCNICA DEL ARTÍCULO

A. Metodología
La metodología propuesta para este proyecto es la orientada a objetos. Ésta consiste en armar módulos basados en componentes y cada componente es independiente del otro. Esto permite que el código sea reutilizable y es más fácil de mantener pues los cambios están localizados en cada componente. Su principal diferencia con la metodología estructurada es que ésta última comprende los procesos como funciones y cada función a realizar por el sistema se descompone en pequeños módulos individuales y finalmente se unen las soluciones para crear la solución al problema.
La ISO (*International Organization for Standardization*) en su norma 12207 define al ciclo de vida de un software como un marco de referencia que contiene las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto software, abarcando desde la definición hasta la finalización de su uso.

Ciclo de vida propuesto: incremental.

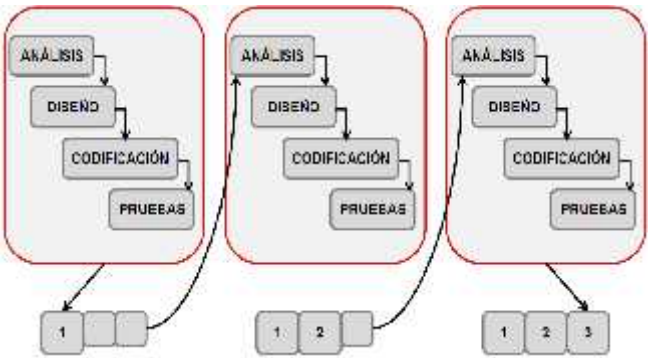

Figura 1 Ciclo de vida de software

Una forma de reducir los riesgos es construir solo una parte del sistema, reservando otros aspectos para niveles posteriores; el desarrollo incremental es el proceso de construcción siempre incrementando subconjuntos de requerimientos del sistema.

En este modelo se desarrolla el sistema para satisfacer un subconjunto de requisitos especificados y en posteriores versiones se incrementa el sistema con nuevas funcionalidades que satisfagan más requisitos. [4]

Características.

- ✓ Combina elementos del modelo de cascada con la filosofía interactiva de construcción de prototipos.
- ✓ Cada secuencia lineal produce un producto operacional con cada incremento de la misma forma que progresa el tiempo en el calendario.
- ✓ El primer incremento es a menudo el núcleo.
- ✓ Como un resultado de evaluación y/o utilización se desarrolla un plan para el incremento siguiente, este proceso se repite hasta llegar al producto completo.
- ✓ Este modelo es particularmente útil cuando la dotación de personal no es suficiente para una implementación completa.
- ✓ Los primeros incrementos se pueden implementar con menos recursos.
- ✓ Si es muy riesgoso desarrollar el sistema completo de una sola vez, entonces debería considerar este modelo.

Ventajas.
• Construir un sistema pequeño es siempre menos riesgoso que construir un sistema grande.
• Al ir desarrollando parte de las funcionalidades, es más fácil determinar si los requerimientos planeados para los niveles subsiguientes son correctos.
• Si un error importante es realizado, sólo la última iteración necesita ser descartada y utilizar el incremento previo.

Desventajas.
• Se presupone que todos los requisitos se han definido al inicio.
• Se requiere de una experiencia importante para definir los incrementos de forma de distribuir en ellos las tareas en forma proporcional.
• Si el sistema a desarrollar es de gran magnitud y se cuenta con un único grupo para construirlo se corre el riesgo que el desarrollo se prolongue demasiado en tiempo.

B. Especificación de requisitos según el estándar de IEEE 830.

Después de la recopilación y verificación de los requisitos, éstos se detallan en un documento, el que ha sido elaborado en colaboración con las profesoras responsables del proyecto de desarrollo del software mencionado.

Esta especificación se ha estructurado basada en las directrices dadas por la última versión del estándar "IEEE *Recomended Practice for Software Requirement Especifications* (ANSI/IEEE) 830-1998"

En términos generales, la segunda versión del sistema SAC deberá proporcionar soporte a las siguientes tareas de control de los alumnos que están cursando las actividades complementarias:

- Mantenimiento de las actividades complementarias.
- Inscripción de alumnos en las actividades.
- Gestión de la participación de los alumnos en las actividades.
- Liberación de alumnos que han acreditado su actividad complementaria.
- Impresión de reportes y constancias de acreditación.
- Acceso a los alumnos para consulta del estado de las actividades complementarias en la que están inscritos.
-

C. Análisis orientado a objetos.

En esta fase se crearon diagramas de casos de uso, de clases, descripciones de escenarios, diagramas de proceso y de actividades, de secuencia, entre otros. A continuación se presentan los diagramas de caso de uso.

Figura 2. Casos de uso principales en el módulo de alumnos

El sistema le solicita al alumno su número de control y su contraseña de acceso.

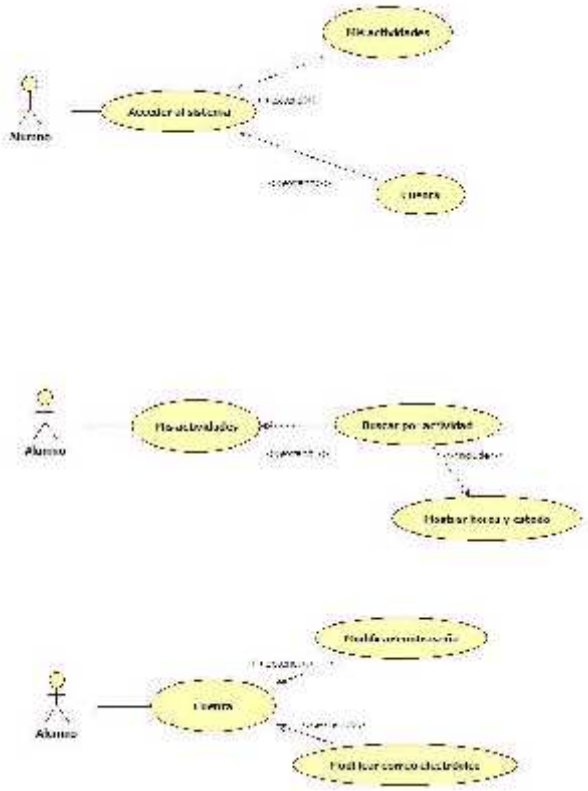

Figura 3. Casos de uso detallado en el módulo de alumnos

Los alumnos pueden modificar su contraseña y correo electrónico si lo desean.

El alumno busca por actividad complementaria y el sistema le muestra las horas que ha acumulado en esa actividad, así como el estado que se encuentra.

D. Modelado de clases

Mediante el Diagrama de Clases de la Figura 4 mostramos el diagrama de clases que incluye a los alumnos.

Diagrama de clases

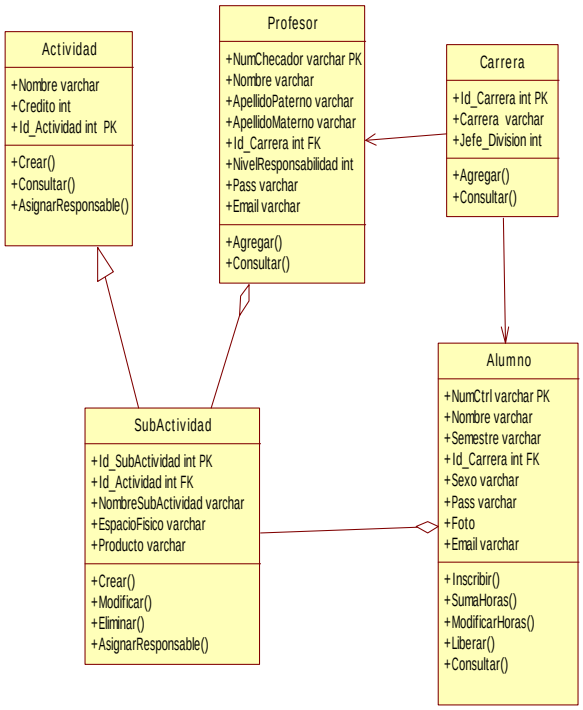

Figura 4. Diagrama de Clases del Sistema

E. Pantallas Sistemas Módulo Alumnos

Figura 5. Esta opción del menú le permite cambiar la contraseña y agregar o cambiar un correo electrónico

Figura 6. Página principal

Figura 7. Consulta de Actividades.

F. Modelo Relacional

Mediante el Modelo Relacional mostrado en la figura 8 podemos visualizar la forma en que se almacenará y relacionará la información manejada por el sistema.

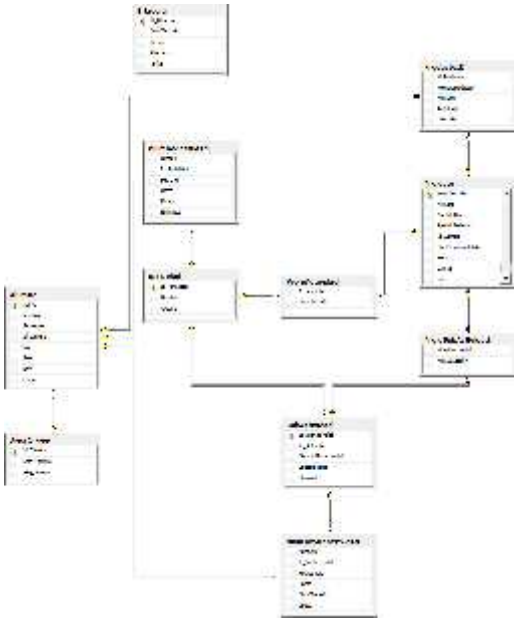

Figura 8. Esquema Relacional de la Base de Datos

G. Diccionario de Datos de las Tablas principales.

Mediante el Diccionario de datos, podemos listar las características lógicas de los datos que vamos a utilizar en

el sistema [5], tales como nombre, descripción, contenido. En la figura 6 se muestra el diccionario de datos de las tres tablas principales del sistema a desarrollar:

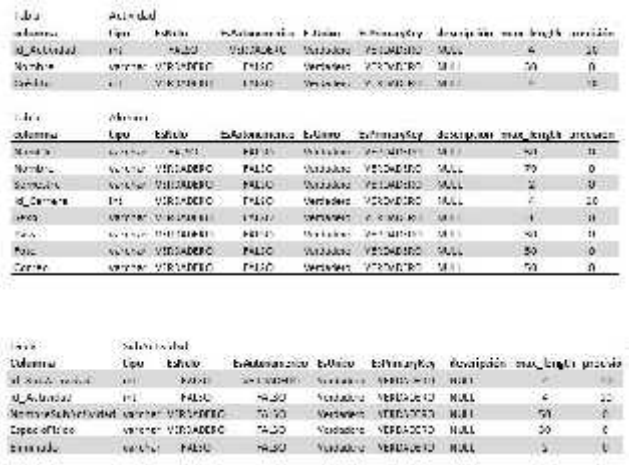

Figura 9 Diccionario de datos de tablas principales del Sistema

H. Impacto o beneficio tecnológico

El principal beneficio que se obtuvo con este proyecto es la sistematización y estandarización de la gestión y control de los alumnos del Instituto Tecnológico Superior de Lerdo inscritos en las actividades complementarias, y el acceso oportuno a la información de los alumnos, profesores responsables, departamentos académicos y divisiones correspondientes.

III. CONCLUSIONES

El proyecto realizado contribuirá y beneficiará de manera importante al control de la acreditación de las actividades complementarias realizadas por los alumnos del Instituto Tecnológico Superior de Lerdo.

El módulo desarrollado permitirá al alumno ingresar al sistema con su nombre de usuario y contraseña, seleccionar una actividad complementaria y consultar cuántas horas lleva acumuladas, el sistema le mostrará el detalle de su inscripción en dicha actividad: la fecha de inicio, la sub-actividad, las horas acumuladas a la fecha y el estado (en curso, completado o liberado), además puede acceder a una interfaz en donde puede cambiar su contraseña y registrar un correo electrónico para recuperación de ésta.

Dado que se realizó un análisis y diseño de sistemas detallado de los requerimientos y las necesidades que se quieren satisfacer, se logró realizar el módulo del sistema con el mínimo riesgo de cambios durante su desarrollo.

IV. AGRADECIMIENTOS

Nuestro agradecimiento es para el Instituto Tecnológico Superior de Lerdo por su apoyo para la continuidad de este proyecto.

V. REFERENCIAS

[1] Vidal, C. L., Schmal, R. F., Rivero, S., & Villarroel, R. H. (2012). Extensión del Diagrama de Secuencias UML (Lenguaje de Modelado Unificado) para el Modelado Orientado a Aspectos. Información Tecnológica, 23(6), 5161. doi:10.4067/S0718-07642012000600007

[2] Schmuller, J. (2001.). Aprendiendo UML en 24 horas. Prentice Hall.

[3] Gamboa, F. G. (2011). Microsoft Project 2010: Microsoft Project 2010. Microsoft.

[4] Pressman, R. S. (2005). Ingeniería del Software Un Enfoque Práctico. Madrid, España: Mc Graw Hill.

[5] Coronel, C., Morris, S., & Rob, P. (2011). *Base de Datos, Diseño, Implementación y Administración*. México, DF: Cengage Learning Editores, S.A.

VI. BIOGRAFÍA

María Guadalupe Flores Luévanos. Nació en Torreón, Coahuila de Zaragoza, México, es Ingeniero en Sistemas Computacionales egresada del Instituto Tecnológico de la Laguna. Torreón, Coah., México (2002). Estudió la Maestría en Administración. Universidad Autónoma de Coahuila. Torreón, Coah., México. (2004).
Actualmente es docente del Instituto Tecnológico Superior de Lerdo, Cd. Lerdo. Durango, México. Sus áreas de interés son el Desarrollo de software en el área de Investigación.

Elda Moreno Núñez. Nació en Gómez Palacio, Durango, México, es Ingeniero en Sistemas Computacionales egresada del Instituto Tecnológico de la Laguna. Torreón, Coah., México (1995). Estudió la Maestría en Administración. Universidad Autónoma de Coahuila. Torreón, Coah., México. (2005). Actualmente es docente del Instituto Tecnológico Superior de Lerdo, Cd. Lerdo. Durango, México. Sus áreas de interés son la ingeniería de software y los sistemas geográficos.

Edna Velia Josefina Solorio Vega. Nació en Torreón, Coahuila de Zaragoza, México, es Ingeniero en Sistemas Computacionales. Egresada del Instituto Tecnológico de la Laguna. Torreón, Coah., México. Estudió la Maestría en Administración. Universidad Autónoma de Coahuila. Torreón, Coah., México. (2008). Ella es docente del Instituto Tecnológico Superior de Lerdo, Cd. Lerdo. Durango, México. Su área de interés es el desarrollo de Proyectos de Software.