UNIVERSIDAD DE CIENFUEGOS "CARLOS RAFAEL RODRÍGUEZ"

CENTRO DE ESTUDIOS DE LA DIDÁCTICA Y LA DIRECCIÓN DE LA EDUCACIÓN SUPERIOR - CEDDES

TÍTULO: FORMACIÓN DE LA HABILIDAD *PROGRAMAR* EN LAS ASIGNATURAS DE PROGRAMACIÓN DE COMPUTADORAS EN LOS PRIMEROS AÑOS DE LA CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TESIS PRESENTADA EN OPCIÓN AL GRADO CIENTÍFICO DE DOCTOR EN CIENCIAS PEDAGÓGICAS

AUTOR: MSc. MILTON MARIDUEÑA ARROYAVE

PROFESOR PRINCIPAL – UNIVERSIDAD DE GUAYAQUIL

TUTORA: DRA. MIRIAM IGLESIAS LEÓN

CIENFUEGOS, 2014

Dedicatoria

A mis padres y hermanos…

A mi esposa Luz Marina y mis hijos: Víctor, José y Rafael, la familia incondicional…

A la Dra. C Miriam Iglesias, quien me hizo crecer con cada una de sus orientaciones…

Al Econ. Rafael Correa, quien con sus sabias decisiones nos enseñara a visualizar el futuro de la investigación científica que apenas comenzamos a construir hoy en la universidad ecuatoriana

Agradecimientos

Estos dos últimos años han sido para mí los más importantes, intensos y fascinantes de mi trayectoria profesional. Tiempo en que he tenido la enorme suerte de conocer y de trabajar con personas que me han ayudado a culminar con éxito este trabajo de investigación.

Mis agradecimientos a todas las personas del CEDDES, aunque el hecho de exponer una lista de personas siempre supone un riesgo de olvidar a alguna de ellas; quisiera hacer un reconocimiento muy especial a la valiosa e incansable ayuda de la Dra. C. Miriam Iglesias León, quien ha sido para mí un auténtico privilegio y honor tenerla como directora. Quizá sea ella la persona que mayor ejemplo me ha dado de rigor, trabajo, búsqueda de la verdad, inquietud por el saber y deseo de abrir horizontes inéditos para la investigación.

A mis compañeros del programa doctoral y colegas de la Universidad de Guayaquil al Dr. Carlos Cedeño N., Dr. José Apolo P. y MSc. Víctor Ballesteros, quienes siempre han confiado en mí y me brindaron amablemente su incondicional respaldo, su tiempo y sus conocimientos.

Y a mi familia..., que me dio la fuerza y el apoyo para seguir adelante, hasta la conclusión de este Doctorado y comprender mis largos días de ausencia, con Amor.

Todo esto no hubiera sido posible sin la guía y protección de mi Dios Todopoderoso.

Síntesis

Las insuficiencias didácticas en el proceso docente educativo de las asignaturas de programación de computadoras en la carrera de Ingeniería en Sistemas Computacionales, detectadas como regularidades de su diagnóstico se concibe un Sistema de Tareas Docentes Integradoras para contribuir a la formación de la habilidad *programar* en los estudiantes de los primeros años de esta carrera. El Aporte Teórico de la tesis está en la concepción de la habilidad *programar* desde la integración de los contenidos disciplinares de la programación de computadoras para construir una síntesis de contenidos esenciales con un mayor grado de generalización en estas asignaturas. El Aporte Práctico radica en la elaboración del Sistema de Tareas Docentes Integradoras para la formación de la habilidad *programar,* donde se apreció el trabajo del criterio de expertos para su concepción, su aplicación en los estudiantes de los primeros años de la Carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil y la Triangulación de los resultados de las Encuestas a estudiantes, profesores y los resultados de la Observación a clases en los cursos de Programación I, II, III y Orientada a Objetos en esta universidad, para su validación.

-Palabras clave: Habilidades de programar,Sistema de Tareas Docentes Integradoras ,

Universidad de Guayaquil

Índice

Índice de Tablas

Introducción

La Educación Superior del presente siglo, demanda procesos innovadores, en los cuales la ciencia ocupa un lugar de privilegio, en tal sentido es necesario, que la universidad perfeccione los escenarios formativos, donde se evidencien las competencias académicas y de gestión, por ello, los profesores respondiendo a esta demanda social, están llamados a garantizar una formación integral, responsable para los nuevos profesionales.

El desarrollo que actualmente ha alcanzado la humanidad tanto en la actividad técnica como social exige de profesionales con un nivel de formación profesional superior.

En la actualidad, la emergente necesidad del manejo de las Tecnologías de la Información y las Comunicaciones (TIC´s), ha posibilitado que la universidad diversifique sus ofertas académicas y responda a la demanda de la sociedad de la información y del conocimiento, a partir de carreras tecnológicas con diferentes enfoques, por lo que resulta evidente que la carrera de ingeniería necesita de ese novedoso enfoque para enfrentar el reto de la diversificación de ofertas académicas.

En el mundo, la formación de los ingenieros está orientada a: planificar, diseñar, programar, construir, mantener y controlar el funcionamiento de máquinas, equipos y sistemas de diversas esferas de la actividad productiva, estos profesionales participan en grupos multidisciplinarios donde desempeñan diferentes tareas como gestión, dirección y coordinación. Estas habilidades son muestra de la necesidad de un elevado nivel de preparación que se sustenta en la calidad del contenido de su formación.

En América, las carreras de ingeniería tienen la tendencia a formar ingenieros de perfil amplio con posibilidades de la especialización en alguna dirección con el empleo del sistema de créditos por opción, manifestándose una fuerte formación socio humanística y respondiendo a objetivos profesionales en función de la demanda de la matriz productiva.

La universidad debe garantizar la formación de ingenieros con conocimientos, habilidades y valores para ponerlos al servicio de la sociedad, el desarrollo de la ciencia y la tecnología, con racionalidad económica, optimización del uso de los recursos materiales, humanos e informáticos, preservando los principios éticos y estéticos sin deteriorar el medio ambiente.

Desde estas premisas, la universidad ecuatoriana está implementando una serie de cambios en su estructura administrativa y académica, con la finalidad de satisfacer en su real dimensión, los requerimientos de la sociedad. Por su parte, el Gobierno de la República del Ecuador, a través de los organismos que controlan y regulan el Sistema de Educación Superior, propone mejorar el nivel de calidad del proceso formativo universitario, basado en estándares de calidad y de pertinencia de cada carrera profesional en su contexto local, regional y global, en los que se evidencia la aplicación de los principios del Plan Nacional del Buen Vivir propuesto por la Secretaría Nacional de Planificación y Desarrollo del Ecuador (SENPLADES, 2013).

A tal efecto, con vista a dar respuestas al encargo social del ingeniero ecuatoriano, se ha diseñado un modelo del profesional, que se caracteriza por una formación sólida en ciencias matemática, física e informática, y un desarrollo de perfil general, capacidad para dar respuesta a los problemas generados en el progreso de su carrera, y la formación de habilidades profesionales que contribuyan a su modo de actuación en la sociedad (UNESCO, 2002).

El modelo del profesional de las carreras de ingeniería vinculadas a las TIC’s en el Ecuador se configura a través de diferentes currículos de formación. Los ingenieros en computación e ingenieros en sistemas computacionales formados en la universidad ecuatoriana poseen modos de actuación invariantes desde la informática, pero con diferencias notables de su

aplicación en el campo profesional y laboral, según la Ley Orgánica de Educación Superior del Ecuador (LOES, 2010).

En vista de la heterogeneidad de currículos de estas carreras profesionales, a partir del 2001 en Estados Unidos de Norteamérica, varias organizaciones científicas y profesionales como son: la Association for Computing Machinery (ACM), la Computer Society (IEEE CS) y la Association for Information Systems (AIS) han desarrollado un documento estándar aceptado mundialmente y reconocido en la actualidad como Computing Curricula (2008).

Este documento intenta normalizar los diferentes currículos en las ciencias informáticas y caracterizar homogéneamente sus lineamientos curriculares en función del modo de actuación del profesional de ingeniería en cinco áreas: Ciencias de la Computación (CS), Sistemas de Información (IS), Ingeniería en Computación (CE), Ingeniería de Software (SE) y finalmente Tecnologías de la Información (TI).

En varias universidades y escuelas politécnicas ecuatorianas según la Secretaría Nacional de Educación Superior, Ciencia y Tecnología (SENESCYT, 2013), se confirma que por las directrices curriculares y los perfiles profesionales, las carreras tecnológicas que se ofertan en estas entidades ecuatorianas, se alinean a la ingeniería en sistemas computacionales, entendidas por la Computing Curricula (2008), como Ingeniería de Software.

Se conceptualiza a este tipo de ingeniero como un profesional que debe poseer habilidades analíticas y de razonamiento lógico, que le posibilita desde la programación de computadoras, a construir soluciones informáticas materializadas como software para la automatización de los procesos organizacionales.

De acuerdo con (López, Whalley, Robbins, & Lister, 2008) los estudiantes de ingeniería en sistemas computacionales, en toda su carrera universitaria desplegarán la formación de habilidades que les permita: ser capaces de resolver problemas mediante la algorítmica y la

programación de computadoras, usar conceptos de sistemas para concebir los problemas de las organizaciones, entender que un sistema se compone en un entorno global: de personas, procedimientos, hardware y software, practicar valores y fuertes principios éticos, poseer una excelente comunicación interpersonal, desarrollar habilidades de trabajo en equipo, demostrar persistencia, creatividad, asunción de riesgos y tolerancia a fallas. Un ingeniero en sistemas computacionales analiza, diseña e implementa soluciones informáticas para mejorar el desempeño organizacional; por lo que los estudiantes en formación deben desarrollar habilidades para comprender los problemas a programar, modelar los procesos y los datos, diseñar algoritmos y codificarlos en lenguajes de programación centrados en la aplicación de tecnologías de la información para ayudar a las personas, grupos y organizaciones a alcanzar sus metas en un entorno competitivo.

El pensum curricular de un ingeniero en sistemas computacionales por el tratamiento del ciclo de producción de software como objeto fundamental de estudio, evidencia una carga horaria muy fuerte en la disciplina de programación de computadoras de los primeros años, en términos de asignaturas como Programación I, Programación II, Programación III y Orientada a Objetos (Anexo No. 01), (Anexo No. 02), todas y desde su interrelación deberían dar respuesta a la formación de estas habilidades que configuran hasta el final de la carrera, el modo de actuación de este profesional Maridueña (2014).

De la revisión del estado del arte, un colectivo de autores A. Corbí (1998), L. Joyanes (2003), L. Rodríguez (2003), M. Fernández (2003), S. Caro (2003), J. Gallardo (2004), C. García (2004) conceptualizan *programar* como la habilidad de escribir las instrucciones o sentencias para que la computadora realice una tarea. Al conjunto de instrucciones lo denominan programa.

El autor analiza el planteamiento anterior y toma en consideración la revisión de los planes de estudio de universidades ecuatorianas y de otros países, el estudio de los balances semestrales de las asignaturas de programación desde 1997 hasta los actuales momentos y la experiencia del mismo durante 17 años de ejercicio docente, lo que permite plantear como principales insuficiencias en relación con la formación de la habilidad *programar* en la carrera de Ingeniería en Sistemas Computacionales, las siguientes:

- El proceso de enseñanza de la programación, se centra con fuerza en el uso de un lenguaje
- Se detecta como regularidad las clases magistrales y la transmisión de conocimientos.
- Se determina con poca relevancia, al proceso de formación de la habilidad *programar*.
- El estudiante desde el primer semestre es llevado a programar en función de la escritura de comandos de un lenguaje de programación específico.
- Existen pocas referencias en documentos institucionales que aborden la formación de la habilidad *programar* desde el análisis del contenido disciplinar de programación de computadoras.
- A partir de los primeros años, no existe un contenido disciplinar con un enfoque de interrelación de las asignaturas de programación de computadoras.
- Los programas de estudio no consideran métodos que vinculen la teoría con la práctica.
- El proceso de enseñanza ha tenido falencias, por un sistema docente incompleto en cuanto a conocimientos teóricos metodológicos.
- La complejidad de conocimientos abstractos propios de esta disciplina, no se aplican en correspondencia con las diferencias individuales de los estudiantes.
- Los estudiantes no articulan ni integran los contenidos actuales con los nuevos contenidos entre las asignaturas de programación de computadoras de dos semestres consecutivos.

Estas insuficiencias han sido analizadas por los profesores en las Juntas de Área y de Carrera de la Universidad de Guayaquil, y se aprecia que la habilidad *programar* no es formada sistemáticamente e interrelacionada en el tratamiento de los contenidos de las asignaturas de programación de computadoras en el proceso educativo desde los primeros años de carrera; lo que afecta a la formación de la habilidad *programar* en los estudiantes.

Por todo lo antes expuesto, se plantea el **problema científico:** ¿Cómo contribuir a la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales?

Como **objeto de investigación:** el proceso docente educativo en las asignaturas de programación de computadoras en los primeros años de la Carrera de Ingeniería en Sistemas Computacionales, y como **campo de acción**: la formación de la habilidad *programar* en las asignaturas de programación de computadoras en los primeros años de esta carrera.

Se plantea como **objetivo:** la elaboración de un Sistema de Tareas Docentes Integradoras para la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales, para mejorar su modo de actuación como profesional.

Como **idea a defender**: la elaboración de un Sistema de Tareas Docentes Integradoras estructurado en un sistema de acciones y operaciones, donde se interrelacionen los contenidos esenciales de las asignaturas de programación de computadoras en los primeros años, de forma teórica y práctica, contribuirá a la formación de la habilidad *programar* en los estudiantes de la carrera de Ingeniería en Sistemas Computacionales.

Como **tareas científicas** se trazan las siguientes:

1. Analizar los referentes bibliográficos sobre la formación de la habilidad *programar* en las asignaturas de programación de computadoras de los primeros años en la carrera de Ingeniería en Sistemas Computacionales, sus tendencias principales y la concepción de la formación de la habilidad *programar* en este profesional.

2. Diagnosticar las principales dificultades que tienen los estudiantes en la habilidad *programar* en los primeros años de Ingeniería en Sistemas Computacionales.

3. Estructurar un Sistema de Tareas Docentes Integradoras, que contribuya a formar la habilidad *programar* en los estudiantes de las asignaturas de programación de computadoras de los primeros años en esta carrera.

4. Validar por criterio de expertos el Sistema de Tareas Docentes Integradoras, basado en un sistema de acciones y operaciones para la formación de la habilidad *programar* en la carrera de Ingeniería en Sistemas Computacionales.

5. Aplicar el Sistema de Tareas Docentes Integradoras en esta carrera y valorar su contribución en la formación de la habilidad *programar* en el estudiante.

Los métodos de investigación que se aplican para alcanzar los objetivos de la investigación son los siguientes:

Métodos teóricos:

El método analítico - sintético presente durante toda la investigación, en el estudio de las fuentes de información para extraer de ellas regularidades y tendencias relacionadas con el proceso de la formación de la habilidad *programar*, a partir de los modelos curriculares de Ingeniería en Sistemas Computacionales, de diferentes universidades ecuatorianas y de otros países.

El método inductivo – deductivo, para realizar el tránsito de lo general a lo particular y viceversa, al establecer el vínculo de los contenidos esenciales entre las asignaturas de

programación de computadoras para concebir la formación de la habilidad *programar* en los estudiantes de los primeros años de la carrera.

El método histórico – lógico, para analizar los programas de estudio de las asignaturas de programación de computadoras de carreras de Ingeniería en Sistemas Computacionales de universidades ecuatorianas y de otros países desde 1997 hasta los momentos actuales para encontrar las regularidades en la concepción de la habilidad *programar*.

Métodos de nivel empírico:

La observación directa, abierta, sistemática y participativa para obtener información sobre la formación de la habilidad *programar*, y a partir de una guía de observación que permita caracterizar el Sistema de Tareas Docentes, así como identificar las regularidades didácticas del sistema de acciones y operaciones.

La aplicación de encuestas a estudiantes y profesores para obtener información sobre la necesidad de la concepción de la formación de la habilidad *programar* y del perfeccionamiento de la habilidad en el Sistema de Tareas Docentes Integradoras.

El análisis documental posibilitó la consulta y estudio de la literatura publicada sobre los programas de carreras de Ingeniería en Sistemas Computacionales de diferentes modelos curriculares, la política educacional del "Plan Nacional del Buen Vivir" explícito en el Registro Oficial de la Ley Orgánica de Educación Superior del Ecuador (LOES), las directrices del Consejo de Educación Superior (CES), el Reglamento de Régimen Académico, boletines de matrícula, balances semestrales, portafolios de seguimiento docente, planes y programas de estudio ecuatorianos y de otros países.

La aplicación del método Delphi para validar el Sistema de Tareas Docentes Integradoras, y su organización para la formación de la habilidad *programar* en los estudiantes de Ingeniería en Sistemas Computacionales.

La triangulación de los resultados de los métodos: para el análisis de los resultados de las encuestas a estudiantes, profesores y la observación a clases, con respecto a la formación de la habilidad *programar* desde los contenidos esenciales de las asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales.

Del nivel estadístico:

Métodos de la estadística descriptiva, para el análisis e interpretación de los datos que se obtienen como resultados de los métodos aplicados y la aplicación de la Teoría del Muestreo Probabilístico para la determinación del tamaño de la muestra. Uso del SPSS para Windows en su versión 15.0, para el procesamiento de datos, la elaboración de las tablas de frecuencias absoluta, relativa, el cálculo de los valores de la distribución Normal Inversa Estandarizada.

El **aporte teórico** de la investigación se manifiesta en la concepción de la formación de la habilidad *programar* desde la integración de los contenidos disciplinares de la programación de computadoras para construir una síntesis de contenidos esenciales con mayor grado de generalización en las asignaturas de programación de computadoras en los primeros años de la formación del ingeniero en sistemas computacionales.

El **aporte práctico** se concreta en el Sistema de Tareas Docentes Integradoras para la enseñanza de la programación de computadoras en los primeros años de la carrera la cual posibilitará la formación de la habilidad *programar* en los estudiantes y contribuirá a mejorar el desempeño como profesional, con la propuesta se enriquece el tratamiento didáctico de la enseñanza de la programación de computadoras en esta carrera.

La **novedad científica:** Lo novedoso de esta investigación está dado en la concepción de la formación de la habilidad *programar* en el ingeniero en sistemas computacionales y la

concepción de un Sistema de Tareas Docentes Integradoras para la formación de esta habilidad, lo que contribuye a mejorar el proceso de aprendizaje de la programación de computadoras, y facilita la formación de esta habilidad para integrar el contenido desde los primeros años, lo que posibilita que el estudiante, reciba sus orientaciones desde las primeras tareas en función de la programación de computadoras como disciplina integrada como un todo, la cual se irá sistematizando de manera gradual en las diferentes asignaturas de programación mediantes ejercicios prácticos, lo que influye favorablemente en la formación de esta habilidad en el modo de actuación del profesional.

La tesis está estructurada en introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos.

En el **primer capítulo** se aborda el estado del arte y la fundamentación teórica respecto a formar la habilidad *programar* en los primeros años de la carrera de Ingeniería en Sistemas Computacionales, así como la concepción de la formación de la habilidad *programar* y el Sistema de Tareas Docentes para contribuir a su formación en los primeros años de carrera.

En el **segundo capítulo** se propone sobre la base del diagnóstico realizado en los estudiantes de los primeros años de la carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil, el Sistema de Tareas Docentes para la formación de la habilidad *programar* con sus indicaciones de manejo para los profesores implicados.

En el **tercer capítulo** se exponen los resultados de los métodos aplicados, planteándose los aspectos esenciales de la validación y aplicación del Sistema de Tareas Docentes Integradoras para las asignaturas de programación de computadoras en los primeros años de la Carrera de Ingeniería en Sistemas Computacionales.

Se exponen las conclusiones finales y las recomendaciones del trabajo.

Capítulo I. Fundamentación teórica sobre el análisis de la formación de la habilidad *programar* en el ingeniero en sistemas computacionales. Su concepción en las asignaturas de programación de computadoras de los primeros años de la carrera profesional.

El objetivo de este capítulo es estructurar el marco teórico de la tesis. Durante su desarrollo se expondrán las tendencias generales de la evolución de las carreras de ingeniería en sistemas, los aspectos significativos en la formación de este profesional, y su contextualización en el ámbito ecuatoriano. Se hace un análisis de las habilidades y la concepción de la formación de la habilidad *programar* en los primeros años de la Carrera de Ingeniería en Sistemas, así como el fundamento y estructuración del Sistema de Tareas Docentes.

1.1 Tendencias de las carreras de ingeniería de las ciencias informáticas.

El nuevo paradigma de las carreras de ingeniería de las ciencias informáticas o ingeniería en informática refleja la forma en que la informática en su conjunto, viene evolucionando para abordar los problemas del nuevo milenio.

De acuerdo con el Informe de la Computing Curricula (2005) en los EE.UU., la ingeniería en informática había cristalizado su status como carrera distinta de la ingeniería eléctrica y asumió un papel principal en lo que respecta a la relación del hardware y el software.

No obstante, la carrera de Ingeniería en Sistemas Computacionales, surge a partir de los importantes desafíos inherentes a la construcción de sistemas de software fiables y asequibles a la sociedad.

Este proceso de evolución ha permitido crear un amplio espectro de carreras informáticas y su oferta académica heterogénea en las instituciones de educación superior. Después de

1990, el aumento de la diversidad de ingenierías en informática implicó que los estudiantes tomen decisiones en un entorno más ambiguo de lo que se presentaba antes de esta década.

En el año 2001, se publicó el primer reporte conjunto de carreras relacionadas con la informática con un volumen para cada una de ellas: Ciencias de la Computación (CS), Sistemas de Información (IS), Ingeniería en Computación (CE) e Ingeniería de Software (SE) según la Computing Curricula (2001).

En octubre del 2005 se presentó a Tecnologías de Información (IT) como una actualización del reporte del 2001 como una quinta carrera que se integrara a las anteriores, Computing Curricula (2005). Cada una de estas carreras presentan disciplinas comunes; no obstante, el tratamiento de la informática tiene diferentes ópticas en función de su campo de acción.

La Ingeniería en Computación (CE) se enfoca en el diseño y construcción de dispositivos digitales. Involucra el estudio de hardware, software y comunicaciones. Su currículo se enmarca en teorías y prácticas de la ingeniería eléctrica y las matemáticas. En esta carrera se estudia el software únicamente para dispositivos digitales y sus interfaces con otros dispositivos. Se hace énfasis en el hardware más que en el software.

La carrera de ingeniería en Tecnologías de la Información (IT) se orienta en integrar las necesidades de computación en las organizaciones. A diferencia del currículo anterior que se enfocaba en el hardware, el perfil de las IT se incorpora en la integración de las tecnologías de la información. Este profesional es responsable de que la infraestructura tecnológica en la organización funcione.

La ingeniería en Ciencias de la Computación (CS) abarca un amplio rango a partir de las teorías y fundamentos algorítmicos en los desarrollos de la Inteligencia Artificial y otras áreas científicas. Estos profesionales son muy teóricos y orientados a la investigación.

El currículo de ingeniería en Sistemas de Información (IS) se enfatiza en la información y las visiones tecnológicas como un instrumento para generar, procesar y distribuirla. Los profesionales de esta área se preocupan por la información que la empresa necesita para alcanzar sus metas. Su estudio se enfoca en los hechos técnicos y los factores organizacionales, para determinar cómo la información y las tecnologías de los negocios, pueden mejorar su ventaja competitiva.

La Ingeniería de Software (SE) se encarga de desarrollar y mantener software que sean confiables y eficientes en concordancia con la satisfacción de los requerimientos de los usuarios (ROBOCODE, 2007). Esta carrera ha evolucionado debido a la creciente importancia e impacto del software en las empresas. Se diferencia de las otras ingenierías por la naturaleza intangible del software y el ámbito discontinuo de su operación. El ingeniero de software integra los paradigmas de la programación de computadoras, para una producción eficiente de software, desde su concepción.

A criterio del autor, desde el análisis de pensum curriculares, el ingeniero en sistemas computacionales es un ingeniero de software (SE) que resuelve la demanda del profesional que es requerido para enfrentar la era de la información, especializándose en la formación de las habilidades necesarias para programar, y gestionar las tecnologías de software adecuadas, que le posibiliten materializar soluciones informáticas, con una visión integral de los requerimientos de las organizaciones.

1.1.1 El proceso de formación de los ingenieros en sistemas. Su expresión en el contexto ecuatoriano.

De la revisión de documentos en (SENESCYT, 2013) donde se presenta la oferta académica de ingenierías en sistemas computacionales en el Ecuador, el autor aprecia que

esta carrera surge como una necesidad debido a que la demanda social en la actualidad exige la formación de profesionales capacitados en el área de la informática.

En este sentido, en (De Graff & Kolmos, 2007) se afirma que la carrera de Ingeniería en Sistemas Computacionales responde a la necesidad de formar profesionales con capacidad para intervenir en las organizaciones con las tecnologías de la información, a fin de proponer, desarrollar implantar soluciones informáticas desde la programación de computadoras, para potenciar el desarrollo social.

Reyes (2009) afirma que la formación de este profesional, se conceptualiza en la universidad como una carrera enfocada al dominio de las ciencias y tecnologías de la información, necesarias para planificar, analizar, diseñar, programar, operar, mantener, evaluar y optimizar sistemas informáticos de diversa índole.

El objetivo común analizado por el autor en las carreras de Ingeniería en Sistemas Computacionales, de las universidades de mayor relevancia en el Ecuador es: Formar profesionales con una sólida formación en las áreas de programación de computadoras, desarrollo de software, sistemas y tecnologías de la información, con el fin de optimizar los recursos computacionales disponibles en las organizaciones; capaces de entender y adaptar las nuevas tecnologías para desarrollar sistemas que apoyen a las áreas funcionales de la organización; profesionales proactivos e innovadores que diseñen, implanten y administren los sistemas mediante las tecnologías computacionales; capacitados para automatizar diversos métodos, técnicas y procedimientos sobre la base de las habilidades de programación, así como en el manejo del diseño y configuración de redes de cómputo, y en la generación de nuevas tecnologías con ética y conciencia de servicio social, que contribuya al desarrollo regional, nacional e internacional.

En (Del Río, 2007) se argumenta que desde el inicio de su carrera, el estudiante de ingeniería en sistemas computacionales debe desarrollar un dominio de conocimientos de programación de computadoras que lo capacite para resolver problemas de su propia carrera y finalmente de su esfera de actuación, por consiguiente según (Kölling, 1999) el proceso educativo debería orientarse a que el estudiante aplique metodologías de análisis, diseño y programación de soluciones informáticas, sin embargo, es necesario que posea además una sólida formación en Ciencias Básicas, Ciencias de la Computación, Tecnologías de la Información, Gestión de las Organizaciones y que el estudiante en estas encuentre un eje transversal temático que le permita adquirir mayor solvencia teórica y práctica en su formación de tal forma, que los contenidos de la carrera tributen con un carácter científico a la formación del profesional.

Por otra parte, en un informe de la Comisión de Evaluación y Acreditación de la Educación Superior en el Ecuador (CEAACES, 2011) se confirma que el núcleo invariante del pensum curricular del ingeniero en sistemas computacionales en el Ecuador comprende las disciplinas matemáticas y física, constituida por dieciséis (16) asignaturas [28% de la carga total], programación de computadoras, software y sistemas, constituida por veinticuatro (24) asignaturas, [41% de la carga total], gestión informática constituida por doce (12) asignaturas, [21% de la carga total] y la disciplina de expresión oral y escrita, constituida por seis (6) asignaturas [10% de la carga total curricular]. (Anexo No. 01)

Esto demuestra que la carga horaria curricular acumulada de las disciplinas que requieren la habilidad *programar* como una invariante funcional en la formación de los estudiantes de ingeniería en sistemas computacionales se aproxima al 90% de la carga curricular total.

No obstante, el informe de la CEAACES (2011) reporta que la formación integral del ingeniero en sistemas es afectada desde el desarrollo de los dominios actitudinales, la base

de conocimientos y el sistema de habilidades que no encuentran un hilo conductor didáctico de interrelación desde los primeros años y cuya insuficiencia se intensifica en la asimilación de nuevos contenidos relacionados con la programación de computadoras en el desarrollo de la carrera, y se demuestra su impacto negativo en el egresado, al incorporarse con deficiencias de su formación a su vida profesional.

1.2. Caracterización de las habilidades. Su fundamentación.

Existe actualmente un consenso general dentro de la comunidad educativa mundial sobre la necesidad de superar el tipo de enseñanza basada en la transmisión de conocimientos para privilegiar en su lugar al desarrollo de habilidades en los estudiantes universitarios. Investigaciones y estudios recientes en (EDUTEKA, 2004) se proponen diversos tipos de habilidades que la educación debe fomentar para que los estudiantes puedan tener éxito en el mundo digitalizado y globalizado en el que se desarrollan. Este planteamiento exige, sin dilaciones, implementar estrategias que contribuyan efectivamente a la formación de esas habilidades planteadas como fundamentales para la educación en el Siglo XXI.

En la mayoría de las habilidades propuestas en el *21st. Century Skills* según EDUTEKA (2004) figuran las habilidades de pensamiento de orden superior entre las que se incluye la habilidad para solucionar problemas mediante computadoras; siendo esta habilidad consustancial a los profesionales de ingeniería en sistemas computacionales, se destaca la importancia de que se seleccionen estrategias didácticas que contribuyan a que los estudiantes las formen en la clase.

Basado en los criterios de Petrovsky (1985) y Zilberstein (2005), el autor realizó un análisis de la concepción de la formación de las habilidades quienes plantean que es "el dominio de un sistema de actividades psíquicas y prácticas, necesarias para la regularización consciente de la actividad, de los conocimientos y hábitos" (p.76).

De manera similar en (Danilov & Skotkin, 1980) se profundiza en el estudio de la formación de la habilidad y en que es "un proceso pedagógico extraordinariamente complejo y amplio: es la capacidad adquirida por el hombre para utilizar creadoramente sus conocimientos y hábitos, tanto durante el proceso de actividad teórica como práctica."

Bajo estas concepciones el autor coincide y considera que la habilidad constituye todo un conjunto de elementos y procedimientos integrados, cuya formación permite una función global en la que el sujeto modifique, adecúe y transforme su actividad, conocimientos y hábitos de acuerdo con sus necesidades.

M. López (1990) y J. Zilberstein (2005) mencionan los requerimientos necesarios para la formación de las habilidades y coinciden con los autores anteriores al considerar que una habilidad constituye "un sistema complejo de operaciones necesarias para la regularización de la actividad y que debe garantizar que los estudiantes asimilen las formas de elaboración, los modos de actuar, las formas de razonar, de modo que con el conocimiento se logre la formación de las habilidades" (p. 561).

G. Fariñas (1995) describe psicológicamente que:

> Un conjunto de habilidades, por su grado de generalización y poder autorregulador de la personalidad, pueden ser colocadas como columna vertebral de cualquier currículo, que pretende encaminar y desplegar el potencial de desarrollo psicológico de la persona y que se denominan Habilidades Conformadoras del Desarrollo Personal (HCDP) porque posibilitan la eficiencia o competencia del individuo, ya sea en la actividad o en la comunicación, en cualquier esfera de la vida porque están en la base de todo aprendizaje y porque son mecanismos de autodesarrollo.(p. 136)

Un colectivo de autores cubanos (2003) define a la habilidad como: "La capacidad de aprovechar datos, conocimientos o conceptos que se tienen, que operen con ellos para la

educación de las propiedades sustanciales de las cosas y la resolución exitosa de determinadas tareas teóricas y prácticas" (p. 235).

M. Silvestre (2000) señala que “(…) como parte del contenido de la enseñanza, la habilidad implica el dominio de las formas de la actividad cognoscitiva, práctica y valorativa”, es decir "el conocimiento en acción" (p.54).

En las acepciones referidas, el autor conjetura que en la habilidad el concepto de su formación se vinculan los aspectos psicológicos y pedagógicos como componentes indisolublemente unidos. Desde el punto de vista psicológico se precisan las acciones y operaciones como componentes de la actividad y desde el punto de vista pedagógico el cómo dirigir el proceso de asimilación de esas acciones y operaciones en correspondencia a su clase o tipo.

M. López (1990) y M. Silvestre (2000) clasifican las habilidades, en generales y específicas según sean parte del contenido de todas las asignaturas o solo de algún tipo en particular. Son habilidades generales: la observación, la descripción, la comparación, la clasificación, la definición, la modelación y la argumentación. Son específicas: el análisis bibliográfico, la interpretación de planos, catálogos, tablas y el uso de determinados instrumentos. Profesores del Instituto de Perfeccionamiento Educacional IPE de la República de Cuba (2003), coinciden y asumen un enfoque semejante de clasificación en habilidades generales de carácter intelectual; y como una categoría diferente a las habilidades docentes generales como la organización, planificación y autocontrol, a esta clase también corresponden las habilidades del uso del libro de texto y otras fuentes de información y las habilidades comunicativas.

El autor acorde con la literatura consultada, destaca que dentro de las habilidades generales están las de carácter intelectual y entre ellas las que favorecen el desarrollo del pensamiento

lógico; así como las denominadas docentes que son las que determinan en gran medida la calidad de la actividad cognoscitiva de los estudiantes, y las habilidades específicas son aquellas que se relacionan con una asignatura concreta, pues el saber es poco probable que pueda materializarse si no es a través de este tipo de habilidad.

Por otra parte, C. Álvarez (2003) considera a las habilidades como: "Un sistema de acciones y operaciones para alcanzar un objetivo" y las clasifica por el modelo del profesional en “básicas, específicas y del ejercicio de la profesión”. El autor asume que una habilidad es profesional si es esencial e indispensable para el modo de actuación del profesional.

En la Teoría de la Actividad, A. N. Leontiev (1982) se plantea que la acción es una unidad de análisis que aparece solamente cuando el individuo actúa. Toda acción se descompone en varias operaciones con determinada lógica y secuencialidad. Las operaciones son pequeñas acciones, son procedimientos, son las formas de realización de la acción atendiendo a las condiciones, o sea las circunstancias reales en las cuales se realiza la habilidad, las operaciones le dan a la acción esa forma de proceso continuo.

En cada habilidad se pueden determinar las operaciones cuya integración permite el dominio por el estudiante de un modo de actuación, una misma acción puede formar parte de distintas habilidades, así como una misma habilidad puede realizarse a través de diferentes acciones, las acciones se correlacionan con los objetivos, mientras que las operaciones se relacionan con las condiciones.

Los conceptos de acción y operación son relativos y no absolutos, lo que en una etapa de la formación de la habilidad interviene como acción, en otra etapa se hace como operación, al proceso donde no existe coincidencia entre motivo (móvil) y el objetivo (representación del resultado) se denomina acción y cuando existe coincidencia se refiere a la actividad, en este

caso a la habilidad. Se puede precisar que el proceso de formación de las habilidades consiste en apropiarse de la estructura del objeto y convertirlo en un modo de actuar.

Atendiendo a los estudios realizados sobre el tema desde el punto de vista didáctico, este autor ha reflexionado en los siguientes presupuestos metodológicos que propician el proceso de formación de las habilidades: planificar el proceso de forma que ocurra una sistematización y consolidación de las acciones; garantizar el carácter activo y consciente del alumno y realizar el proceso solventando el aumento progresivo del grado de complejidad y dificultad de las tareas y su correspondencia con las diferencias individuales de los estudiantes.

El autor coincide con los criterios en (Talízina, 1987) y sustenta que:

- El profesor al seleccionar los contenidos de la enseñanza, debe tener presente no solo el sistema de conocimientos que en correspondencia con los objetivos deben ser asimilados por los estudiantes sino también los tipos de acciones generales y específicas o particulares, ya que los conocimientos solo pueden ser asimilados cuando los estudiantes realizan algunas acciones con los mismos. Solo se puede dirigir el proceso de aprendizaje mediante la ejecución de las acciones que los estudiantes deben realizar para apropiarse de los conocimientos y la asimilación de cualquier contenido.

- Las habilidades tienen una estructura integrada por tres aspectos fundamentales: En primer lugar el conocimiento específico, segundo el sistema operacional específico (acciones) y finalmente los conocimientos y las operaciones lógicas.

- Toda acción o actividad humana ya sea mental, perceptual, motora, posee una composición de elementos que pueden ser considerados como invariantes; y al respecto apunta interesantes aspectos del proceso de enseñanza- aprendizaje sobre el estudiante: el objetivo cuyo cumplimiento se satisface mediante la habilidad, el objeto sobre el que recae

la acción del estudiante, un motivo para realizar la actividad, un sistema de operaciones o procedimientos para realizar la acción, la base orientadora para la acción que determina la estructura de dicha acción, los medios para la realización de la actividad, las condiciones en que se realiza la actividad y el resultado de la acción.

- Para garantizar adecuadamente la asimilación de los conocimientos, las habilidades deben responder a tres criterios básicos: la adecuación de las habilidades a los objetivos de la enseñanza, las habilidades seleccionadas deben profundizar los conocimientos esenciales y el proceso de formación de las habilidades debe apoyarse en las leyes de la asimilación.

Se asume el estudio de las habilidades partiendo del planteamiento del objetivo, en correspondencia con el mismo determinar la esencia de los conocimientos que deben ser asimilados por los estudiantes y tener en cuenta el carácter activo y consciente del proceso.

En el sistema de acciones específicas para formar la habilidad hay que destacar dos tipos de acciones: las específicas para apropiarse del conocimiento comprender y fijar; y aquellas que le permitan operarlo.

De manera que las operaciones lógicas son las que permiten la asimilación y aplicación de los conocimientos adquiridos por los estudiantes. Manifiesta L. A. Corona (2006) que no se puede desarrollar una habilidad determinada sin la presencia de las acciones mentales u operaciones lógicas tales como el análisis, la síntesis, la comparación, la abstracción, la generalización, además de las acciones de evaluación en cada habilidad.

La formación de las habilidades en la Educación Superior, según (Cañedo, 2004) exige la necesidad de atender las diferentes formas de organización de la docencia a la luz de un nuevo enfoque, en el cual no siempre el punto de partida sea la clase teórica, el laboratorio, el taller, la clase práctica; sino que la formación de la habilidad puede partir también de una situación problémica, mediante la implementación individual o grupal de proyectos de

producción de software a un tamaño de escala acorde con la sistematización de contenidos en ese momento, que lleve al estudiante y al profesor a la búsqueda de información especializada, a la indagación a expertos, a la recolección y procesamiento de datos que induzcan a reflexionar acerca de las formas de solucionar el mismo y su posterior fundamentación teórica en las clases de ejercitación, talleres, sesiones de laboratorios.

El criterio que toma el autor para la formación de las habilidades se fundamenta, precisamente, en la ruptura de aquellas maneras de pensar tradicionales y en su lugar situar a los estudiantes ante situaciones problémicas como parte del proceso docente educativo.

1.2.1 La concepción de la habilidad *programar* en las asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales.

Las primeras referencias sobre programar computadoras son asignadas a Ada Lovelace, a la cual se le atribuye por varios autores como (Laboda, Galimany, Peña, & Gual, 1985).

En la literatura científica revisada, (Zhu & Zhou, 2009) y G. Levine (2001) llegan a un consenso que programar es una actividad basada en teoremas matemáticos, los que se aplican en el dominio de la técnica para plantear algoritmos y buscar la solución más rápida a los problemas profesionales planteados. .

La disciplina de programación de computadoras de acuerdo con (Joyanes, Rodríguez, & Fernández, 2003) tiene como propósito la solución de problemas mediante el empleo de computadoras, la cual se identifica con los paradigmas de la programación estructurada, y orientado a objetos. En los ingenieros en sistemas computacionales, esta disciplina resulta de gran importancia para la formación de las habilidades profesionales vinculadas a la producción de software.

Las actualizaciones realizadas a los planes de estudio de esta disciplina, sostiene E. Cassola (2004) que han evolucionado desde el estudio de lenguajes de bajo nivel, lenguajes intérpretes y compiladores hasta lenguajes de alto nivel, estudiándose en la actualidad los fundamentos de la programación modular estructurada y finalmente los paradigmas modernos orientados a objetos.

En (Perry, 2001) se afirma que el área de programación como disciplina curricular dota a los estudiantes de la metodología de construcción de algoritmos, entendidos como estructuras básicas de proceso, de la forma más eficiente de manejar tipos de datos simples y estructurados, así como de las estructuras lógicas de datos que permitan hacer un uso más óptimo del espacio de memoria, de minimizar los tiempos de acceso, y lograr formas más efectivas para operar las estructuras de almacenamiento de datos, los cuales son procedimientos aplicables a cualquier lenguaje de programación de nueva generación.

Los fundamentos de programación estructurada que aporta la disciplina, proveen al estudiante de un soporte teórico metodológico para la comprensión del nuevo paradigma, la programación orientada a objetos, como forma particular de enfrentar la solución de problemas en términos de clases y objetos, así como en la interacción con dichos objetos en un entorno visual, mediante la respuesta a los posibles eventos que puedan ocurrir.

Teniendo en cuenta estos presupuestos para impartir los conocimientos debe predominar como enfoque didáctico, el enfoque problémico, pues tiene como objetivo central la resolución de problemas haciendo énfasis en la creación de nuevas situaciones problémicas, es decir, mediante problemas crear la necesidad del nuevo conocimiento de la programación de computadoras, objeto de estudio.

Un colectivo de autores a nivel internacional como L. C. Green (1991); J. Beane (1991), S. M. Drake (1991); R. Fogarty (1993); J. Torres (1995); J. Gimeno (1995) y (Vars & Beane,

2001) han aportado criterios que permiten reconocer que hoy día la integración de los contenidos en el currículum posee amplia perspectiva.

A criterio del autor, integrar es articular partes para conformar un todo. De ahí que podemos concordar que integrar los contenidos de la disciplina de programación de computadoras de los primeros años, consiste en enlazarlos armónicamente de manera que con la unidad de las partes se contribuya al mismo objetivo visto como un todo. Sería, coincidiendo con J. P. Fiallo (2004) "…ver la integración como un momento de organización de los contenidos de las disciplinas, es una etapa para la interacción, que solo puede ocurrir en un régimen de coparticipación, reciprocidad, mutualidad (condiciones esenciales para la efectividad de un trabajo interdisciplinar)" (p.178).

La integración de los contenidos ofrece varias ventajas de acuerdo con E. Ortiz (2006) provee: "(…) conexiones entre las áreas curriculares, ofrece al estudiante oportunidades de aplicar habilidades y contenidos aprendidos, provoca la participación activa de los estudiantes en experiencias reales relevantes, motiva y reta a los estudiantes y logra una comprensión más profunda del contenido" (p.123).

De esta forma, el autor deduce, que este enfoque como procedimiento didáctico, debe ser aplicado según los objetivos y contenidos de cuatro asignaturas que forman la disciplina de programación de computadoras de los primeros años y propiciar un aprendizaje activo, desarrollador, independiente e integrador.

Por esto es muy importante que el profesor tenga en cuenta estos elementos, resaltando en ello los ejes temáticos más relevantes de cada asignatura de modo que el estudiante pueda asimilar los contenidos que se tratan, siendo estos insuficientes ya que estas dificultades han persistido históricamente en las carreras de Ingeniería en Sistemas Computacionales, y en la actualidad, las mismas fueron detectadas en diagnósticos, observaciones a clases, y

balances semestrales, convirtiéndose en la problemática central de los estudiantes de esta carrera.

Teniendo en cuenta que las asignaturas de Programación I, II, III y Orientada a Objetos una a una, no están exentas de las dificultades antes mencionadas y que se detecta desde el primer semestre del pensum curricular, sin dejar de mencionar que es la base fundamental para la formación de la habilidad *programar* hasta el 4to semestre de dicha carrera, razón por la cual el autor se plantea elaborar un Sistema de Tareas Docentes para elevar el aprendizaje de los contenidos de estas asignaturas en los estudiantes de los primeros años de la carrera de Ingeniería en Sistemas, de la Universidad de Guayaquil.

En estas asignaturas, el estudiante se dotará del núcleo de conocimientos y habilidades más esenciales de la programación estructurada y orientada a objetos, los cuales propician la resolución de problemas vinculados con el manejo de lenguajes de programación en los que se recorren los diferentes paradigmas.

Plantea A. Carballosa (2004) "(…) que una enseñanza integrada, agrupa contenidos de varias disciplinas y se forma así una nueva unidad de síntesis interdisciplinaria con mayor grado de generalización" (p.65).

Por tal efecto, el autor destaca que los contenidos que se implementan en estas asignaturas, deben tener un carácter interdisciplinario de todo el currículo y se recomienda al profesor que debe tomar algunas bases gnoseológicas sobre las asignaturas del nivel horizontal del currículo y propiamente a nivel vertical de la disciplina, para que los estudiantes sean capaces de interiorizarlos y vincularlos a los diferentes contenidos que se trabajarán en cada una de las unidades del programa de estudio y su mejor comprensión a partir de la resolución computarizada de problemas.

A partir de las técnicas algorítmicas básicas de la programación sostienen R. L. Constable (2008) y (Zhu & Zhou, 2009) que deben crearse las bases teóricas necesarias para que el estudiante pueda fundamentar la formación de la habilidad *programar* permitiendo de esta forma desarrollar proyectos investigativos de ingeniería de software, que pongan de manifiesto el grado de desarrollo alcanzado en cuanto a esta habilidad, así como su creatividad e independencia como reflejo del modo de actuación de ingenieros en sistemas computacionales que pretenden desarrollar estas asignaturas.

En (López, Whalley, Robbins, & Lister, 2008) se considera que los objetivos generales de la disciplina de programación de computadoras deben ser: caracterizar los diferentes paradigmas de la programación a partir del análisis de los elementos que los tipifican y sus metodologías inherentes, resolver problemas de diferentes contextos aplicando las diferentes metodologías, teniendo en cuenta los recursos que brindan los paradigmas de programación; propiciar la formación de habilidades profesionales en el proceso de la resolución de problemas en correspondencia con los principios que coadyuvan a la actividad profesional y promuevan el aprendizaje independiente con un enfoque interrelacionado e integrador.

En el análisis realizado por los autores citados anteriormente se aprecia que los mismos no trabajan la programación de computadoras de forma integrada, sino que la orientan de forma asistemática, ante esta situación, el autor propone que la concepción sea de forma interrelacionada en los contenidos desde las primeras acciones de la Programación I hasta la Programación Orientada a Objetos.

El carácter integrador que da la relación del contenido de las asignaturas de programación de computadoras, dota al estudiante desde la Programación I de los elementos esenciales del paradigma de la programación estructurada, que sirven de fundamento teórico para la

Programación II donde se manejan conceptos más complejos de la programación estructurada que tributan junto con los fundamentos de la Programación I a la comprensión de los contenidos de la Programación III, en los que se configuran estructuras de datos con mayor complejidad de abstracción, que junto a la integración de contenidos de la Programación I y II estructuran una plataforma base de contenidos, para la siguiente asignatura de programación de la disciplina en los dos primeros años de carrera, que presenta un nuevo paradigma, la Programación Orientada a Objetos. La fundamentación de los paradigmas de la Programación Estructurada y la Orientada a Objetos constituye la base teórica, para el resto de asignaturas de programación de computadoras de la Carrera de Ingeniería en Sistemas Computacionales.

Por otra parte, el estudiante que transita por estas asignaturas además de integrar contenidos, debe consolidar los conocimientos de cada una de ellas, que se aplican desde la práctica con un enfoque problémico, incremental y de interrelación a partir de una sistematización de las etapas de construcción de un programa informático, que se inicia en la concepción del problema, el diseño del algoritmo, la codificación y la depuración del programa. Para lograr esta integración se referencia la necesidad de utilizar en la formación de esta habilidad, lo relacionado con la enseñanza problémica.

A. Ortiz (2005) reconoce que en la enseñanza tradicional de programación de computadoras se busca esencialmente la formación de un pensamiento empírico, el estudiante al aprender a programar es un receptor pasivo y el docente al enseñar es activo, ante esta regularidad, M. Majmutov (1983) criticó este tipo de enseñanza que por lo general le ofrece al estudiante, los conocimientos elaborados, sin comprender cómo fue el proceso de búsqueda y construcción teórica que lo condujo a esos conocimientos.

En este sentido, el autor coincide con M. Majmutov (1983) quien considera a la enseñanza

problémica, como la actividad del profesor encaminada a la creación de un sistema de situaciones problémicas, y a la dirección de la actividad de los estudiantes en la asimilación de conocimientos nuevos.

P. Poirier (1997) afirma, que "una situación problémica es una situación concreta, que describe a la vez el contexto lo más real posible y la tarea frente a la cual el estudiante debe poner en orden los conocimientos y procedimientos necesarios, para el desarrollo y la demostración de su habilidad" (p.64).

Se comparte la definición anterior, se asume a las acciones que deberá ejecutar el estudiante en los diferentes contextos o modos de actuación profesional, las cuales serán a partir de una situación inherente al objeto de la profesión.

A tal efecto, la actividad docente desde los primeros años de la carrera de ingeniería en sistemas computacionales, debe estar encaminada a la asimilación productiva de conocimientos de programación de computadoras, mediante la percepción de las explicaciones del profesor en las condiciones de una situación problémica, lo que propiciará la formación de la habilidad *programar* a partir de la integración de contenidos de la programación estructurada y orientada a objetos.

Este autor concibe la formación de la habilidad *programar* como la actividad que realiza el estudiante de forma interrelacionada y práctica de los contenidos esenciales de la lógica de programación en sus paradigmas estructurado y orientado a objetos y que propicia la solución de problemas desde la informática.

Esta definición de la formación de habilidad *programar* posibilita desde el proceso de enseñanza - aprendizaje, un conjunto de acciones y operaciones, facilitadas por el profesor para el manejo de la computadora bajo el efecto de los paradigmas de la programación estructurada y orientada a objetos, que facilitarán el desarrollo de las habilidades del

pensamiento lógico de los estudiantes, mediante el tratamiento didáctico de las invariantes de contenidos de programación en las asignaturas de los primeros años de la carrera de Ingeniería en Sistemas Computacionales.

En los autores revisados, no se precisa de una concepción en cuanto a la didáctica de la formación de la habilidad *programar*; evidenciándose la necesidad de ella. No obstante, en este trabajo se presenta desde la pedagogía, la concepción de la formación de esta imprescindible habilidad para la función del ingeniero en sistemas computacionales; con su estudio desde los primeros años en las asignaturas de programación de computadoras como habilidad básica para la función del profesional a través de interrelación de los contenidos interdisciplinares del currículo, lo que la identifica como una habilidad profesional.

1.2.2 La habilidad *programar* como invariante de habilidad en la dimensión de contenidos en la Carrera de Ingeniería en Sistemas Computacionales.

La formación y desarrollo de las habilidades profesionales en una carrera universitaria ha sido objeto de investigación en las últimas décadas, derivándose diferentes alternativas didácticas, entre ellas: (Talízina, 1987), (Fuentes & Cols, 1997), (Álvarez, 2002), (Álvarez, 2003), (Iñigo, 2005) y (Lapido, 2009).

Establecer una invariante de habilidad significa para N. Talízina (1987) determinar entre todas las habilidades la que resulta fundamental o esencial, o que en calidad de invariante debe aparecer en el contenido de las disciplinas y llegar a ser dominado por el estudiante.

L. Joyanes (1996) afirma que "(…) la habilidad *programar*, los conceptos básicos de algoritmos, estructura de datos y objetos, así como el modo de aprender a resolver problemas mediante computadoras, para cualquier currículum de Ciencias o Ingeniería Informática es de vital importancia para su carrera en la universidad y su ejercicio profesional (…)" (p.87).

En (Iñigo, 2005) se infiere que múltiples son las habilidades que debe asimilar un ingeniero en sistemas computacionales y todas de una forma u otra, se interrelacionan a través de un hilo conductor, un elemento rector que organice esta relación. Este elemento lo constituye el invariante de la habilidad *programar*.

De acuerdo con (Lapido, 2009), el invariante de habilidad expresa una generalización esencial de habilidades y como habilidad generalizada integra y subordina a otras más simples. Su valor facilita al estudiante el proceso de apropiación del objeto de la profesión, pues le permite enfrentar y resolver de manera consciente los problemas complejos que se relacionan con el ejercicio de la profesión aplicando dicha invariante.

Según los aportes de (Álvarez, 2003) se asume que del Modelo del Profesional o Perfil Profesional, se derivan las habilidades y disciplinas que conforman el pensum curricular (disciplinas Básicas, Básicas Específicas y de la Profesión), el mismo responde a los problemas profesionales y a las habilidades que el egresado debe dominar para enfrentar con éxito el ejercicio de la profesión.

Las disciplinas o áreas Básicas, contienen aquellas ciencias o ramas del saber que, sin identificarse con el objeto de la profesión ni con alguna de sus partes, sí poseen contenidos imprescindibles para poder operar con dichos aspectos profesionales.

Las disciplinas Básicas Específicas, contienen los campos de acción de la profesión, es decir, son aquellas que instruyen al alumno en los fundamentos científicos y tecnológicos de un aspecto esencial de la profesión.

Las disciplinas del Ejercicio de la Profesión, contienen las esferas de actuación del profesional. Son aquellas donde el estudiante aprende los aspectos científicos y tecnológicos de la profesión y además penetra en las relaciones sociales, humanas y administrativas inherentes a su profesión.

El objetivo de las asignaturas de programación de computadoras en el currículum de un ingeniero de sistemas computacionales en sus primeros años, no es únicamente que el estudiante aprenda a escribir un programa de computador. El desarrollo de estos contenidos según: A. Corbí (1998); (Copi & Cohen, 2000); S. Caro (2003); (Gallardo & García, 2004); J. Iranzo (2005), (Scaffidi, Shaw, & Myers, 2005) y S. L. Del Río (2007) generan una gran cantidad de operaciones mentales en los estudiantes: ellos deben aprender a entender un problema: *abstraer, modelar, analizar*, a plantear soluciones efectivas: *reflexionar sobre una abstracción, definir estrategias, seguir un proceso,* aplicar una metodología: *descomponer en subproblemas, a manejar lenguajes de programación* para expresar una solución: *codificar, respetar una sintaxis*, a utilizar herramientas que entiendan esos lenguajes: *programar, ejecutar, depurar,* a probar que la solución sea válida: *entender el concepto de corrección y de prueba*, a justificar las decisiones tomadas: *medir, argumentar,* siendo estas habilidades básicas de un ingeniero en sistemas computacionales.

A partir de lo expresado en los epígrafes anteriores, se introduce la idea fundamental, de que *programar* es una habilidad invariante del ingeniero en sistemas computacionales, ya que es de necesario dominio en las disciplinas del currículo informático y sin ella, es complejo resolver problemas técnicos vinculados a la producción de software.

1.2.3 La Unidad de la Teoría con la Práctica y la Sistematización de la enseñanza como principios didácticos en la estructuración de la habilidad *programar* en los primeros años de las asignaturas de programación de computadoras en la Carrera de Ingeniería en Sistemas Computacionales.

Los principios didácticos constituyen una condición básica para organizar el proceso de enseñanza – aprendizaje en cualquiera de sus niveles estructurales. Se encuentran ideas con

fundamentación psicológica y pedagógica sobre esto, desde J. Rousseau (1712 – 1778), J.E. Pestalozzi (1746 – 1827), J.A. Comenius, K.D. y Ushinski (1824 – 1870).

Estos autores plantearon ideas sobre los principios al considerar que los mismos se expresan en el quehacer docente, y propician las conclusiones adecuadas para orientar el proceso y garantizar la dirección sistemática del mismo.

Lothar Klinberg (1978) considera a los principios como aspectos generales de la estructuración del contenido organizativo metódico de enseñanza que se originaron de los objetivos y de las leyes que los rigen objetivamente V. V. Davidov (1986) los definen como los métodos de aplicación de las leyes de la enseñanza, en correspondencia con los fines de la educación y la enseñanza.

N. S. Savin (1972) llama principios de la enseñanza a los principios rectores de partida que determinan la enseñanza y el estudio en correspondencia con los objetivos de la educación y las regularidades del proceso de enseñanza.

Al analizar estos criterios clásicos, se detecta como idea fundamental y frecuente que los principios de la enseñanza son guías, posiciones rectoras, postulados generales, los cuales están presentes en el proceso de enseñanza – aprendizaje, el estudio de los principios didácticos es un tema polémico desde la diversidad de clasificaciones y la nominación de estos por diferentes autores que han abordado los mismos en la pedagogía contemporánea, no obstante se precisan aquellos que reflejan determinadas regularidades de la enseñanza y contribuyen a analizar el proceso docente.

En la escuela cubana existen numerosos autores que trabajan los principios didácticos y asumen a estas categorías como postulados para estructurar el proceso en todos sus niveles, se destacan (Labarrére & Valdivia, 1988) entre la comunidad pedagógica que utilizan a estos postulados en sus obras científicas.

Por la importancia que reviste en la formación de las habilidades, el análisis de estos postulados generales como sistema, se asume a los principios enunciados por (Labarrére & Valdivia, 1988), tales como: principios del carácter educativo de la enseñanza, el carácter científico de la enseñanza, la Asequibilidad, la Sistematización de la enseñanza, la Relación entre la teoría y la práctica, la atención a las diferencias individuales dentro del carácter colectivo del proceso, la unión de lo concreto y lo abstracto y la solidez en la asimilación de los conocimientos.

En el análisis de formación de la habilidad *programar* que se plantea en este trabajo, se seleccionan dos principios didácticos como base para la formación de la habilidad *programar* en los estudiantes, el principio de Unidad de la Teoría y la Práctica y la Sistematización de la Enseñanza, estos principios posibilitan la dirección de la formación de los estudiantes a aprender a *programar*, desde la propia práctica y para la práctica, con un enfoque de integración de todos los contenidos de programación de computadoras que va desarrollando en su formación, esta peculiaridad; hace que el autor fundamente estos principios para la concepción de la habilidad *programar* en la formación de los ingenieros en sistemas computacionales.

Al respecto se precisa en (Collazo, 2005) que la teoría en la enseñanza se basa en el sistema de contenidos que deben aprender los estudiantes, para que este proceso sea duradero consciente y logren mayor grado de asimilación se deben estructurar actividades prácticas de programación en las que los estudiantes se deben involucrar.

Estas actividades prácticas se dan en forma de situaciones problémicas reales o simuladas, que expone y explica el profesor en clase y los estudiantes resuelven en forma de talleres de ejercitación. En los laboratorios de programación se implementan proyectos reales de producción software de pequeña y mediana escala, en función de los convenios de

vinculación con la colectividad y los relacionados con las necesidades administrativas de la propia carrera y se practica en investigación, en lo que respecta a la búsqueda de información especializada sobre el uso de nuevos lenguajes de programación y nuevas tecnologías de información, la recolección y procesamiento de datos, el levantamiento de las especificaciones funcionales del software, las pruebas de modelos y la simulación de prototipos de software.

La implementación de estos proyectos con carácter integrador vinculados a la producción real de software, involucra los conocimientos procedentes de diferentes asignaturas, con la implicación personal de los estudiantes por alcanzar un conocimiento íntegro sobre un fenómeno en un contexto determinado, en cuya explicación se demuestre la utilización de conocimientos de varias de ellas. A través de este trabajo de investigación, el autor entiende, que las características de este tipo de proyectos están dadas en que:

- Presuponen la integración, sistematización y transferencia de conocimientos a otras áreas.
- Revelan las relaciones que se pueden establecer entre las asignaturas de la disciplina de programación de computadoras y las posibles relaciones interdisciplinares.
- Potencian el desarrollo de principios y valores conforme a los ideales de la sociedad.
- Requieren del protagonismo de los estudiantes y exigen en particular, la implicación personal, un pensamiento reflexivo, creativo y la valoración personal y colectiva de estos del proceso y su propia ejecución.

Por otra parte, el principio de la Unidad de la Teoría con la Práctica en la formación del ingeniero en sistemas computacionales propicia un equilibrio, que garantiza una preparación científica y la formación de las habilidades profesionales, procesos cognitivos y de actitudes positivas, concretados en la sensibilidad, la formación de criterios propios, participación productiva, participativa y creativa que le permitirá una formación holística.

Este principio se considera una exigencia de la formación científica educativa de la enseñanza, pues los orienta a utilizar los contenidos para resolver los problemas de la vida con valor científico.

El análisis de este principio presupone en la formación de las habilidades la exigencia de actividad práctica para que la habilidad se logre, con un enfoque de interrelación de los contenidos que necesita el estudiante para resolver los problemas dado el carácter interdisciplinario que tienen los contenidos en la actualidad, por ello es factible precisar la importancia de asumir para la formación de la habilidad el principio de la Sistematización de la Enseñanza.

El principio de la Sistematización de la Enseñanza tiene como esencia que toda actividad docente sea consciente de una planificación con secuencia lógica, en esta sentencia se revisa a (Comenius, 1983) en *Didáctica Magna* y el análisis de varios autores contemporáneos estudiosos del tema, los cuales sostienen que este principio se deriva de las leyes de la ciencia que señalan que la realidad es una, y forma un sistema, y que solo se divide en partes de acuerdo con el objeto de estudio que impone la didáctica a las asignaturas y disciplinas en las carreras, para su dosificación en tiempo y espacio, con vista a la gradual asimilación de los contenidos. La invariante de contenido es el carácter de interrelación en la formación de las habilidades.

El tratamiento de la habilidad *programar* debe ser sistemática, cada una de las asignaturas tributan a que el estudiante aprenda a programar, debe estructurarse en función de los conocimientos y las habilidades, con sentido y significación para los estudiantes (valores) previamente planificados de manera que el estudiante los integre como parte de un todo.

La relación de los contenidos de cada asignatura de programación de computadoras le provee al estudiante el carácter integrador siguiendo la estructuración del esquema de

construcción de un programa informático que a su vez se alinea al proceso de formación de la habilidad *programar* como un todo. (Anexo No. 04)

La contextualización del principio del vínculo de la Teoría con la Práctica y el principio de la Sistematización de la Enseñanza en la formación de la habilidad *programar* en las asignaturas de programación de computadoras exigen que en el proceso de formación de esta habilidad se desarrollen de manera consciente las generalizaciones teóricas que permitan a los estudiantes operar con sus conceptos, leyes, establecer nexos y relaciones para que la habilidad *programar* adquiera sentido y significado en los estudiantes, y esta pueda formarse como invariante de contenido en su formación profesional.

El autor asume la unidad de la teoría con la práctica y la sistematización, como principios que integran los contenidos de las asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales y que posibilite al estudiante apropiarse del método algorítmico de solución de problemas basado en computadora y contribuye a su dominio, desde el concepción y análisis del problema, diseño del algoritmo, traducción y depuración del programa informático (Anexo No. 04), considerando estas como las invariantes para desarrollar la habilidad *programar*.

Los fundamentos didácticos de esta interrelación, están sustentados en garantizar el carácter plenamente activo y consciente del estudiante, en planificar la estructura de esta interrelación en las asignaturas desde los primeros años de la carrera, con una sistematización y consolidación de las acciones y operaciones seleccionadas y organizadas de acuerdo con los niveles de asimilación del conocimiento: reproductivo, productivo y creativo, en correspondencia con las diferencias individuales de los estudiantes.

La estructura didáctica de la interrelación se fundamenta en la interacción de los componentes del proceso docente educativo para contribuir a la formación de la habilidad *programar*.

1.3 El Sistema de Tareas Docentes: Su concepción, características, tipos y estructura, para la formación de la habilidad *programar* en las asignaturas de programación de computadoras de los primeros años en la Carrera de Ingeniería en Sistemas Computacionales.

Para A. N. Leontiev (1982) la categoría actividad se presenta como una organización estructural que refleja en lo fundamental la interacción del hombre con el mundo de los objetos. Este autor plantea: "(...) la actividad no es una reacción, así como tampoco un conjunto de reacciones, sino que es un sistema que posee una estructura, pasos internos y conversiones, desarrollo" (p.95). Se induce de lo anterior que la actividad es un proceso complejo y conforma un sistema que como tal posee una estructura.

En el trabajo doctoral de M. Iglesias (1998) se expresa que, la actividad es una característica esencial en el hombre. Para desarrollar su relación con el medio, en ella se forman y desarrollan los procesos psíquicos y las cualidades de su personalidad.

En el estudio de la estructura general de la actividad hay que considerar que de la misma forma que el concepto de motivo se relaciona con el concepto de actividad, así también el concepto de objetivo se relaciona con el concepto de acción.

A. N. Leontiev (1982) considera que las acciones mediante las cuales se realiza la actividad, constituyen sus componentes fundamentales y denomina acción "al proceso que se subordina a la representación de aquel resultado que habrá de ser alcanzado, es decir, el proceso subordinado a un objetivo consciente" (p.46).

Según los puntos de vista de diferentes autores A. N. Leontiev (1982), H. Brito (1984), N. Talízina (1987) y J. López (1989) la actividad existe a través de las acciones y esta a su vez se sustenta en las operaciones.

Otro aspecto esencial en el análisis de la estructura de la actividad lo constituyen las condiciones, entendiéndose por estas las circunstancias en las cuales se realizan los instrumentos de la actividad y los conocimientos, hábitos y habilidades del individuo.

M. Iglesias (1998) sostiene que la acción transcurre en dependencia de las condiciones en que se deba alcanzar el objetivo o fin. Las operaciones son la forma de realización de la acción, o son los procedimientos para cumplirlas.

Las investigaciones realizadas por N. Talízina (1987) han demostrado que fuera de la actividad cognoscitiva los conocimientos no pueden ser adquiridos. Como esté organizada y estructurada la habilidad para desarrollar la actividad cognoscitiva de los estudiantes en el transcurso de su aprendizaje, será el desarrollo alcanzado por ellos y su capacidad para resolver independientemente los problemas de su profesión.

En la literatura pedagógica y psicológica revisada aparece que la clasificación y definición de las tareas es un problema abierto. Al respecto V. Okon (1968) subraya que en la vida y en la escuela se encuentran muchas tareas, cuya solución requiere únicamente una actividad mecánica que solo no contribuye al desarrollo de la independencia del pensamiento, sino que frena ese desarrollo.

La opinión de que cualquier tarea hace avanzar el pensamiento se expresa en los trabajos de muchos pedagogos como M. Danilov y N. Skatkin (1980) quienes señalan que el fundamento de toda tarea, lo constituye la contradicción entre lo que se tiene y lo que la persona quiere alcanzar.

El pedagogo C. Álvarez (2002), plantea que la tarea docente constituye la célula de la actividad conjunta profesor - estudiante y es "(…) la acción del profesor y los estudiantes dentro del proceso, que se realiza en ciertas circunstancias pedagógicas, con el fin de alcanzar un objetivo de carácter elemental de resolver el problema planteado a estudiar por el profesor" (p.64), en los componentes en que se desarrolla el proceso de formación.

Argumenta C. Álvarez (2003) que las tareas deben tener su estructura y la primera condición para la activación del pensamiento es el contacto del sujeto - estudiante con el objeto - condiciones

En trabajos de los autores D.B. Elkonin (1986), V.V. Davidov (1986), C. Álvarez (2001), E. Moltó (2009), se precisa que el proceso docente educativo tiene un elemento estructural o célula del mismo e identifica así a la tarea docente.

El término tarea tiene varias acepciones en la literatura revisada. Dentro de la gran variedad de criterios de clasificación y tipos se encuentra en el estudio de la actividad y su estructura que A. N. Leontiev (1982) la define como: "el objetivo de la actividad que se desarrolla en una situación concreta" (p.58), se induce que la tarea constituye una aspiración científica a alcanzar para la solución de un problema en el proceso de la actividad.

M. Iglesias (1998) afirma que en el planteamiento de la tarea debe manifestarse la contradicción entre lo conocido y lo desconocido, como motor impulsor para su solución. Cada tarea se determina por los objetivos y su duración en la enseñanza, por el carácter del contenido y por las condiciones en que se realiza. La solución de la tarea, implica la transformación del propio sujeto actuante y en algunos casos la del objeto de estudio. La tarea tiene un aspecto intencional *el objetivo* y uno operacional *formas y métodos*.

Se plantea que dicho proceso se desarrolla de tarea docente en tarea docente, hasta que se alcance que el estudiante se comporte del modo esperado. De esta forma el proceso referido

en la educación superior, se manifiesta por una sistematización de tareas docentes, que se desarrollan desde la primera actividad docente hasta alcanzar el objetivo propuesto.

V. V. Davidov (1986) argumenta que la actividad cognoscitiva está encaminada a la solución de las tareas docentes y que cuando se les propicie a los estudiantes, estos se introducen en una situación que requiere de orientación para solucionar todas las variantes posibles de los datos. Para este autor la característica fundamental de la tarea docente es el dominio de los estudiantes de los procedimientos teóricamente generalizados de solución, de determinada clase de tareas.

Del análisis de la literatura revisada por el autor, se destaca la clasificación de las tareas como un problema actual. L. Ya Lerren pone como fundamento de la clasificación de las tareas la base del contenido; D.N. Bogolovleski plantea la necesidad de elaborar una clasificación que tenga como fundamento las particularidades de soluciones de tareas-problemas; D.M. Grishin (1965), divide las tareas docentes en cognoscitivas, prácticas y creativas, pero lo hace atendiendo a sus rasgos externos pues toma como fundamento para su clasificación el conocimiento o desconocimiento del resultado; V. Okón (1968), subdivide las tareas cognoscitivas docentes en problémicas y no problémicas y destaca diferentes tipos de tareas problémicas.

El pedagogo alemán H. Weck clasifica las tareas de acuerdo con las ramas del saber. El holandés J.P. Van de Geer presenta una clasificación de problemas y tareas a partir de sus rasgos externos e internos, pero son poco aceptables los principios de su clasificación, pues a decir de J. L. Rubinstein (1977) "(…) se da la separación de lo que hay que hacer" (p.14).

A criterio del autor, cualquier tipo de tarea que los profesores escojan desarrollará una habilidad en los estudiantes, así como la complejidad de la misma y su combinación favorecerán el desarrollo de esta, teniendo en consideración el período académico, las

características de los estudiantes, las características de los profesores y las condiciones en que se desarrolla el proceso docente educativo.

En referencia al aporte de M. Iglesias (1998) las etapas didácticas que debe utilizar el profesor para garantizar el cumplimiento de las tareas docentes son: orientación de la tarea docente, ejecución de la tarea docente por parte del estudiante, cooperación entre el alumno y el profesor consulta y control de la tarea docente.

Las tareas docentes tienen definidas las acciones a realizar por el estudiante para alcanzar el objetivo; para llevar a cabo la ejecución de la habilidad que se pretende formar en ellas. Las operaciones precisadas en cada tarea docente expresan las condiciones necesarias para que el estudiante lleve a cabo las acciones contenidas en la tarea y con ello, el logro del objetivo, la formación de la habilidad.

A tal efecto, sostiene L. A. Corona (2010) que en cada tarea se definen los elementos a considerar en la evaluación, en la valoración del cumplimiento del objetivo; evaluación en la que se le confiere gran peso a la autoevaluación del estudiante y la coevaluación de este en el seno del grupo.

La autoevaluación y coevaluación se facilitan al contar los estudiantes con una apropiada representación mental de la habilidad a formar a partir de los objetivos, las acciones y las operaciones en que se estructuran las tareas docentes.

(García, González, & Soto, 2012) concuerdan con la definición de tarea docente de C. Álvarez (2002) cuando plantea que "(…) es la célula del proceso docente educativo, en ella hay un conocimiento a asimilar, una habilidad a desarrollar, un valor a formar. Mediante el cumplimiento de las tareas docentes el estudiante se instruye, desarrolla y educa" (p.37).

Mediante las etapas de la orientación, la comunicación del profesor de las diferentes tareas

y nivel de asimilación de las habilidades que debe manifestarse en el estudiante, se van formando convicciones, valores y sentimientos en los mismos.

El proceso se desarrolla de etapa en etapa en las que el profesor va utilizando un conjunto de procedimientos o modos que posibiliten, mediante la comunicación, la incorporación activa del estudiante con vistas al dominio del objetivo. El estudiante en su actividad desarrolla sus propios métodos y procedimientos va adquiriendo las habilidades planificadas y va manifestando su independencia al ejecutar las diferentes tareas, las cuales los preparan para ejecutar los métodos que por sí solo entiende que son necesarios utilizar para solucionar nuevos problemas, más difíciles, llegando a los niveles de asimilación de carácter creativo.

La estructura de la tarea docente se asume como referencia la propuesta de M. Iglesias (1998) en su tesis doctoral, según la cual la estructura está constituida por los siguientes elementos: objetivos, acciones, operaciones y evaluación.

En la elaboración de las tareas se determina un objetivo generalizador para cada una de ellas; este objetivo expresa didácticamente una habilidad a adquirir por el estudiante. Esta habilidad, a su vez, constituye una acción a ejecutar por el estudiante en la toma de decisiones de *programar*; el conjunto de estas habilidades conforma las invariantes funcionales de la habilidad *programar*. Por ello, la sistematización por el estudiante de las habilidades a formar con cada una de las tareas docentes que conforman el sistema de tareas, conduce a la formación de la habilidad *programar*.

De acuerdo con (Zardón & Rojas, 2013) la tarea docente debe ser concebida como un sistema que permita establecer relaciones entre las diferentes acciones y operaciones que promuevan la concepción de la actividad para los propósitos a alcanzar, pudiendo influir tanto en la instrucción, en el desarrollo y en la educación del mismo

Las tareas docentes deben manifestar el objetivo, el contenido y las condiciones para su realización, en su dependencia de la base orientadora que necesita el alumno ajustándose al nivel de dependencia cognoscitiva; estas deben propiciar la búsqueda de contenido y su eficiente utilización en la vida para lograr la estimulación adecuada de los procesos lógicos y operacionales del alumno; por ello el docente debe ser preciso en el conocimiento que se necesita, en los métodos, en los procedimientos que deben emplearse, así como en las acciones y las operaciones que se deben realizar.

De los aportes de autores referenciados en los párrafos y epígrafes anteriores, respecto a la actividad, las acciones, la tarea docente sus tipos y el modelo de estructuración didáctica, el autor asume el término de Sistema de Tareas Docentes, como el sistema de acciones y operaciones seleccionadas y organizadas que le permitan al estudiante apropiarse de la interrelación de la Teoría y la Práctica con la Sistematización de las invariantes en los contenidos de las asignaturas de programación de computadoras en los primeros años de la carrera para desarrollar la habilidad *programar*.

1.3.1 El carácter integrador de la Tarea Docente y la Concepción del Sistema de Tareas Docentes Integradoras para la formación de la habilidad programar en las asignaturas de programación de computadoras de los primeros años de la carrera de Ingeniería en Sistemas Computacionales.

Sostiene L. A. Corona (2010) que la integración existente entre las tareas de Familiarización, Reproducción, Producción y Creación que componen el Sistema de Tareas Docentes "(…) es evidente, además de compartir acciones y operaciones y utilizar en ellas las mismas habilidades de base, no se puede realizar una tarea si no se ha resuelto el problema implícito en la tarea precedente en el sistema" (p.12).

Al ser utilizado como método para la formación de la habilidad *programar*, el propio método algorítmico de programación de computadoras, se determinan respectivamente acciones y operaciones que integran las demás tareas del sistema, en los diferentes momentos de formación de la habilidad *programar*. Estas tareas docentes se integran y expresan las invariantes funcionales de la habilidad *programar*, de esta manera, la lógica con la que el estudiante ejecuta las tareas docentes está determinada por la lógica del método de enseñanza de la habilidad, el método algorítmico implicado en la lógica formación de la habilidad *programar*.

La interrelación de los contenidos esenciales de las asignaturas de programación de computadoras basada en los diferentes niveles de asimilación, permite identificar que el tránsito del estudiante por las distintas tareas del sistema y su interrelación materializada en tareas integradoras, le aporta con los conocimientos, habilidades y valores necesarios para ejecutar la habilidad *programar*.

En las condiciones en que se desarrolla el proceso docente educativo de las asignaturas de Programación I,II,III y Orientada a Objetos, analizando sus principales regularidades el autor propone la clasificación de las tareas atendiendo al nivel de asimilación de los contenidos según los objetivos de la enseñanza, es decir, el estudiante bajo la dirección del profesor ejecuta los procedimientos teóricos necesarios como parte del método que lo acerca al objetivo, por lo que de acuerdo con esta concepción se proponen: tareas de Familiarización; de Reproducción; de Producción, de Creación y como resultado de la interrelación entre ellas, basado en los principios de la Unidad de la Teoría con la Práctica y el de la Sistematización de la enseñanza, se genera un nuevo tipo de tareas que implican un enfoque integrador.

Por ser el método de enseñanza para la formación de la habilidad *programar* el propio método algorítmico de la programación de computadoras, el Sistema de Tareas Docentes Integradoras, como concreción didáctica del método de enseñanza, será utilizado por el estudiante sistemáticamente en su interacción con su objeto de trabajo de programación, en todas las disciplinas de la carrera. El tránsito del estudiante por las distintas asignaturas de su carrera, también lo va dotando del cuerpo de conocimientos correspondiente a los campos de acción de la profesión que cada una de estas disciplinas representa; es decir va completando la base gnoseológica necesaria para la ejecución de la habilidad *programar* en su formación de ingeniero en sistemas computacionales.

Con el propósito de solventar las deficiencias que presentan los profesores de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales en la concreción de la dirección del proceso de integración de los contenidos de las asignaturas Programación I, II, III y Orientada a Objetos de esta carrera se plantea asumir la tarea integradora como la vía a utilizar para lograr que los futuros egresados sean capaces de aplicar los contenidos de programación de computadoras en la solución de problemas profesionales. Esto es posible si se concibe la tarea como un proceso flexible, participativo, creativo, abierto, sistemático y con un carácter de interrelación donde se integren los contenidos de programación de computadoras declaradas como básicas en el Perfil Profesional en torno a las clases de tareas que debe ser capaz de resolver el egresado de esta carrera.

La tarea integradora deberá asumirse desde una posición teórica que responda a la educación científica y pertinente, que demanda el enfoque integrador del Perfil Profesional que establece desde los primeros años el pensum curricular de la Carrera de Ingeniería en Sistemas Computacionales.

Reconocer el origen social de los procesos psicológicos superiores, requiere que el profesor contextualice el aprendizaje del estudiante, y exige que las dimensiones que se consideren en la concepción, desarrollo y evaluación de las tareas integradoras se enmarquen en los diferentes modos de actuación del futuro profesional y que sean ajustadas acorde con el entorno social en que se desarrolla el proceso.

En este sentido se tiene en cuenta que en el proceso de integración de contenidos de las asignaturas de programación de computadoras, se potencia la formación del egresado en términos de conocimientos, habilidades, hábitos, capacidades, cualidades, convicciones y actitudes, lo cuales se alcanzan mediados por la interacción que el estudiante de esta carrera, establece con las características y potencialidades de las asignaturas de programación de computadoras en su contexto socio-laboral, en la que la comparación con sus propias cualidades y potencialidades propician el desarrollo, formación y desempeño profesional en la solución de problemas de la Ingeniería en Sistemas Computacionales.

Si el proceso pedagógico ocurre a partir del tránsito que se produce de lo externo a lo interno en las asignaturas de Programación I, II, III y Orientada a Objetos, como un proceso dialéctico de lo inter.-psicológico a lo intra-psicológico; requiere de la actividad meta-cognitiva del estudiante, y lo pone en condiciones de desarrollarse por sí mismo, al descubrir su lógica individual; entonces las tareas se convierten en un elemento de vital importancia para la formación de un ingeniero en sistemas computacionales capacitado para enfrentar los nuevos retos que impone la sociedad del conocimiento y la información.

La tarea docente al constituirse como el componente integrador del proceso de enseñanza-aprendizaje puede presentarse con diferentes niveles de complejidad en que el estudiante identifica y busca la posible solución a problemas de su esfera de actuación profesional, las cuales pueden ser modeladas a través de situaciones problémicas.

Según (Advine & García, 2005) en el VI Seminario Nacional para Educadores, la Tarea Integradora se define como:

> Una situación problémica estructurada a partir de un eje integrador y tareas interrelacionadas (…) su resultado es la formación de saberes integrados expresados en nuevas síntesis más generalizadoras de los objetos y procesos de la práctica educativa con un enfoque integrador, lo que implica un modo de actuación (p.43)

Se comparte la definición de estos autores, pero a los efectos de esta investigación se toma como eje integrador el enfoque problémico vinculado a los paradigmas de programación estructurada y Orientada a Objetos, presentes en los contenidos de las asignaturas de programación de computadoras de los primeros años.

Estas tareas integradoras se deben caracterizar porque el contenido, las condiciones y las exigencias hacia el estudiante estén relacionados con el objeto de la profesión. En ellas se puede materializar la profesionalización en dependencia de su vínculo con la actividad del profesional, como también se concreta la fundamentación al exigir operar con las invariantes de la habilidad *programar*, ante nuevas variantes, de modo que la lógica de la programación sirva de fundamento para encontrar la solución. La sistematización viene dada por la utilización del sistema de contenidos, por los nexos entre ellos y con la práctica.

Estas tareas constituyen una vía efectiva para que los estudiantes adquieran conocimientos con un mayor grado de generalización, solidez, aumentando la posibilidad de aplicarlos a situaciones nuevas incentivando el desarrollo de un pensamiento independiente, que una vez graduados se desempeñen con pertinencia en el ejercicio profesional.

Cuando las tareas integradoras se conciban como un sistema es posible lograr que estas contribuyan significativamente a la formación integral de los futuros profesionales.

Esta concepción asumida favorece, asimismo, los procesos de comprensión, de solidez, de generalización y evita la atomización de los conocimientos.

1.5 Conclusiones parciales del Capítulo.

A través del estudio de los referentes bibliográficos sobre la formación de habilidades del ingeniero en sistemas computacionales, se analizaron las tendencias principales y se concibe la formación de la habilidad *programar* en las asignaturas de programación de computadoras de los primeros años de esta carrera, como la actividad que realiza el estudiante de forma interrelacionada y práctica de la lógica de la programación estructurada y orientada a objetos, implicada en la dimensión de los contenidos esenciales en esta disciplina de los primeros cuatro semestres de la carrera (Anexo No. 01), (Anexo No. 02), mediante un sistema de acciones y operaciones indispensables para la solución de los problemas vinculados a la producción de software.

Se precisa la formación de la habilidad *programar* sobre la estructura basada en los principios didácticos de la Unidad de la Teoría con la Práctica y el principio de Sistematización de la Enseñanza, mediante un Sistema de Tareas Docentes Integradoras soportado en un sistema de acciones y operaciones, con énfasis en los niveles de asimilación implicados en el enfoque problémico mediante tareas de Familiarización, Reproducción, Producción y Creación, a partir de la integración de los contenidos esenciales de las asignaturas de Programación I, II, III, Orientada a Objetos, y la construcción de una síntesis de contenidos esenciales que se integran con un mayor grado de generalización en cada momento de transición del estudiante en el proceso de formación de la habilidad, implicado en el paso de las invariantes: Concepción del problema, Diseño del algoritmo, Traducción del algoritmo a un lenguaje de programación y su Depuración. (Anexo No. 04)

Capítulo II: Propuesta del Sistema de Tareas Docentes Integradoras para la formación de la habilidad *programar* en las asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales.

En este capítulo a partir de la caracterización del objeto, se presentan los fundamentos del Sistema de Tareas Docentes Integradoras para desarrollar en los estudiantes la habilidad *programar* sobre la base de la interrelación de contenidos esenciales de las asignaturas de programación en los primeros años de la carrera de Ingeniería en Sistemas Computacionales, implicada en las invariantes de la concepción del problema a programar, el diseño del algoritmo solución, traducción del algoritmo a un lenguaje de programación y su depuración; así como las Etapas Didácticas de Orientación, Ejecución y Evaluación propuestas para el efecto.

2.1 Caracterización del proceso docente educativo en las asignaturas de programación de computadoras en los primeros años en la carrera de Ingeniería en Sistemas Computacionales.

La carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil tiene como misión, visión y un pensum curricular que establece las vías y formas de configurar el Perfil Profesional. Se planifica y organiza la carrera con una estructura horizontal basada en semestres y vertical distribuidas en disciplinas y/o asignaturas. Esta carrera se caracteriza por formar un ingeniero de perfil amplio, cuyo objetivo está dirigido fundamentalmente a la producción de software. Los campos de acción en los que se desempeña y las esferas de actuación se presentan en (Anexo No. 01) y (Anexo No. 03)

Las premisas fundamentales que caracterizan a esta carrera están armonizadas con los rasgos esenciales que plantean las tendencias actuales en la formación de estos profesionales, enunciadas en el epígrafe 1.1.1 del presente trabajo.

En los primeros años de la carrera, la disciplina de programación de computadoras constituida por las asignaturas de: Programación I, II, III y Orientada a Objetos, ubicadas consecutivamente desde el primer al cuarto semestre en equivalencia a los dos primeros años de la malla curricular (Anexo No. 01), se enseña al estudiante los fundamentos de programación desde los paradigmas de la programación estructurada y de orientación a objetos, para resolver problemas a través del computador y donde el estudiante emprende el desarrollo de habilidades como el análisis, la síntesis, la comprensión e interpretación de situaciones problémicas. Las asignaturas de esta disciplina constituyen una línea base desde la dimensión de los contenidos para aproximadamente el 90% del total de asignaturas de la carrera de Ingeniería en Sistemas Computacionales.

Programación I (Anexo No. 02)

La asignatura programación I tiene como objetivo marcar las bases en los conceptos de programación estructurada, conociendo los inicios de los lenguajes de programación, aprendiendo y luego dominando la estructura básica de un programa y de esta forma desarrollar software sencillos.

Los estudiantes de ingeniería en sistemas computacionales deben formar sólidas bases en cuanto al conocimiento de los ambientes de programación y estructuras generales de los lenguajes de programación que le permitirán al estudiante entender y desarrollar con claridad cada una de las aplicaciones que se le solicite previo al razonamiento lógico y secuencial buscando soluciones más eficaces y eficientes.

Esta asignatura en el pensum curricular de ingeniero en sistemas computacionales tiene como objetivos: Manejar los tipos de datos predefinidos y hacer uso de cada uno de ellos; definir lo que se entiende por constantes y variables y su manera de declararlas y utilizarlas; conocer los diferentes operadores relacionales y boléanos, así como trabajar y hacer programas como expresiones que hagan uso de los operadores; aprender a manejar las estructuras de control, con decisiones y operadores relacionales, contadores y acumuladores; definir las funciones propias para la resolución de problemas; conocer el uso de las estructuras de datos conocidas como arreglos, de una dimensión y aprender la conveniencia de usarlo en un programa.

Programación II (Anexo No. 02)

Esta asignatura de la disciplina de programación de computadoras pertenece al segundo semestre complementa la enseñanza de la programación estructurada de la Programación I, proporciona los conocimientos necesarios de Estructuras de Datos fijas, el manejo de archivos, la utilización adecuada de memoria física y la comunicación básica de datos.

Los conocimientos adquiridos en esta asignatura permitirán al alumno gestionar adecuadamente la memoria física de un computador, así como métodos de organización de datos en archivos contribuyendo con los fundamentos de manejo de información, que permitan asentar las bases del manejo de datos en dispositivos de almacenamiento.

En el desarrollo del curso, se codifican los programas en lenguaje C++, un lenguaje de programación con características de híbrido ya que permite manejar los dos paradigmas de programación: estructurada y orientada a objetos.

Programación III (Anexo No. 02)

En esta asignatura de programación del tercer semestre, se aspira a que el estudiante: Conozca el uso de funciones mediante llamadas por referencia; aprenda el concepto de

recursividad; utilice punteros en arreglos y manejo de cadenas de caracteres; utilice las estructuras de datos; aprenda el paso de estructuras de datos a funciones llamadas por valor y por referencia; gestione la asignación y liberación de memoria mediante el uso de TDA´s (Tipos de Datos Abstractos); utilice el TDA tipo Lista, el TDA LIFO (Pilas), el TDA FIFO (Colas); gestione archivos secuenciales y aleatorios y utilice los fundamentos para el uso de protocolos de comunicación.

Programación Orientada a Objetos (Anexo No. 02)

Es la cuarta asignatura de la disciplina de programación corresponde al cuarto semestre equivalente al segundo año de la carrera de Ingeniería en Sistemas Computacionales, introduce al estudiante en un paradigma nuevo de programación que tiene como base a los objetos y sus interacciones para la construcción del diseño y desarrollo de soluciones informáticas. En primera instancia se abordan las nociones básicas de programación orientada a objetos, sus ventajas, su diseño y las diferencias frente a la programación estructurada. Se fundamenta los conceptos con el diseño de soluciones basadas en un lenguaje que soporte las características de la orientación a objetos (Lenguaje C++/JAVA). Al final de la asignatura se desarrollan temas avanzados como interfaces gráficas, creación de controles y acceso a base de datos.

El aprendizaje de este nuevo conjunto de técnicas permitirá al estudiante de una forma muy natural, desarrollar su creatividad al momento de proponer una solución informática a un problema determinado.

A tal efecto, la programación de computadoras desde la dimensión de sus contenidos en las asignaturas del pensum curricular, debe tributar a la formación integral del estudiante de Ingeniería en Sistemas Computacionales y a partir de las exigencias planteadas en el proceso de formación de este profesional, corresponde a la programación con fines

profesionales asumir la mayor parte de esta responsabilidad teniendo en cuenta que la Programación Orientada a Objetos es una asignatura terminal del área/disciplina, que funciona como asignatura integradora y de especialización de la carrera.

El estilo fragmentado, inconexo, asistemático en que se realiza la estructuración de contenidos en las asignaturas de programación de computadoras en los primeros años de esta carrera, genera como consecuencia que el estudiante tenga una visión segmentada y alejada de la real dimensión de lo que significa *programar* para una futura producción de software. El estudiante en este escenario le da más importancia a los elementos del lenguaje de programación que al proceso en sí de construir un programa.

Para tal efecto, el autor plantea, que para contribuir al objetivo de la integración entre las asignaturas Programación I, II, III y Orientada a Objetos en la formación de ingenieros en sistemas computacionales se debe considerar a la tarea docente como el medio integrador idóneo, pues según C. Álvarez (2003) *"es la célula del proceso pedagógico."*(p.78), esta no se concibe, ni se enfrenta de forma aislada, sino formando parte de un sistema.

Basado en las conclusiones de los epígrafes anteriores, se presenta la propuesta de un Sistema de Tareas Docentes Integradoras que propicie la generación de una síntesis de contenidos esenciales, alrededor de los ejes temáticos de las asignaturas de programación de computadoras, a partir de la interrelación de las invariantes de contenidos, principalmente de las habilidades con enfoque integrador, para contribuir a la formación de la habilidad *programar* desde los primeros años de la carrera de Ingeniería en Sistemas Computacionales.

2.2 Fundamentos psicopedagógicos del Sistema de Tareas Docentes Integradoras para la formación de la habilidad *programar* en las asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales.

Los fundamentos psicológicos de L. S. Vigotsky (1978), P. Galperin (1982) y N. Talízina (1987) hacen énfasis en el enfoque histórico- cultural, sus premisas emergen en las exigencias de la teoría general de la dirección y las regularidades del proceso de asimilación y el desarrollo de la independencia cognoscitiva, fundamentos que sustentan las premisas generales de esta investigación. Vygotsky enfatiza en el proceso de la cultura humana el cual transcurre a través de la actividad como proceso que mediatiza la relación entre el hombre y su realidad objetiva. A través de ella el hombre modifica la realidad y se forma y transforma a sí mismo, precisando que "(…) el carácter activo de los procesos psíquicos. El punto nodal del desarrollo social y humano lo constituye el concepto actividad, con su atributo esencial: la actividad productiva transformadora". Introduce el concepto de zona de desarrollo próximo y zona de desarrollo actual como el conjunto de actividades que el sujeto puede realizar por sí mismo sin la guía y ayuda de otras personas.

Por su parte L. S. Vigotsky (1978) y P. Galperin (1982), profundizan en la Teoría de la Actividad y proponen un modelo más detallado de la formación por etapas de las acciones mentales, donde el sujeto transita durante el proceso de aprendizaje por diferentes etapas de una forma planificada, que conllevan a una interconexión entre los procesos de exteriorización e interiorización de las acciones, fundamentos que sustentan la proyección curricular del ingeniero en sistemas computacionales.

El autor asume este enfoque, pues asimila al aprendizaje como un proceso de apropiación histórico - social mediante la actividad del sujeto y en la comunicación con otras personas.

No obstante, se comparte con J. Zilberstein (2005) en que la didáctica del proceso de enseñanza desarrollador se fundamenta en el estudiante como centro del proceso, y la solución de problemas mediante tareas docentes, como la vía fundamental para su desarrollo.

Se coincide con (Jenkins, 2001) y se concibe en esta tesis a la tarea docente, como un conjunto de acciones que se realizan, con el fin de alcanzar un objetivo, donde cada acción se desarrolla atendiendo a las condiciones y a la personalidad del sujeto como un todo, tanto lo referido a su esfera motivacional-afectiva como cognitivo-instrumental, o sea, implica considerar también los recursos propios que posee la persona para actuar.

En cuanto al término tarea integradora como expresa C. Álvarez (2002) que: …" el planteamiento de la tarea con una concepción integradora constituye un factor decisivo para la instrucción a la vez que para el desarrollo y la educación de los estudiantes, ya que favorece el compromiso de estos con su realidad incitándoles a una participación más activa, responsable y crítica en ella" (p.65).

Se comparte la definición de este autor teniendo en cuenta que la tarea integradora constituye la vía para que los futuros egresados integren los contenidos de varias asignaturas.

Por otra parte, se identifica la etapa de la formación de una habilidad como centro del trabajo que se desarrolla y es aquella que comprende la adquisición de conocimientos de los modos de actuar, cuando, bajo la dirección del profesor el estudiante recibe la Base Orientadora para la Acción (BOA) sobre la forma de proceder. La formación de las habilidades depende de las acciones, de los conocimientos, hábitos, valores conformando todo un sistema que contiene la habilidad.

La BOA según N. Talízina (1987), se presenta atendiendo a tres características fundamentales, por su carácter generalizado, según su plenitud y según el modo de obtención. Las diferencias entre estas tres características sirven de fundamentos para clasificar los tres tipos de BOA, estos fundamentos se precisarán en la función del Sistema de Tareas Docentes Integradoras, las cuales se estructurarán desde el tipo de tareas de familiarización, hasta la producción o la creación desde las condiciones que se establezcan para las mismas en función de los objetivos.

El primer tipo se caracteriza por una composición incompleta de la BOA y se avanza muy lentamente, con un gran número de errores.

El segundo tipo se caracteriza por la existencia de todas las condiciones necesarias para un cumplimiento correcto de la acción, brindándole al sujeto en forma preparada y particular que sirve para la orientación solo en el caso dado. La acción formada es más estable que en el primer tipo, no obstante, la esfera de la transferencia de la acción está limitada por la similitud de las condiciones concreta de su cumplimiento.

La base orientadora para la acción del tercer tipo se caracteriza por tener una composición completa, están representados en su forma generalizada y concreto, la elabora el sujeto independientemente por medio del método de generalización, le son inherentes no solo la rapidez y el proceso, carente de faltas, sino también una gran estabilidad y amplitud del traslado.

Para N. Talízina (1987) la elaboración del Sistema de Tareas Docentes Integradoras se puede precisar que el proceso de formación de las habilidades consiste en apropiarse de la estructura del objeto y convertirlo en un modo de actuar, en un método para el estudio del objeto, donde juega un papel preponderante la asimilación del conocimiento.

Es decir comenzar por el planteamiento del objetivo, en correspondencia con el mismo determinar la esencia de los conocimientos que deben ser asimilados por los estudiantes y tener en cuenta el carácter activo y consciente del proceso.

Para determinar el trabajo con las habilidades a desarrollar en un área o asignatura, es fundamental esclarecer las habilidades generales y específicas.

Los fundamentos psicológicos del Sistema de Tareas Docentes están sustentados en la selección y organización de las acciones y operaciones en el proceso de formación de las habilidades profesionales del ingeniero en sistemas computacionales.

Las acciones se correlacionan con los objetivos y las operaciones lo hacen con las condiciones que acontece el estudiante en la actividad por alcanzar los objetivos de la tarea.

Las habilidades se forman y desarrollan por la vía de la ejercitación, mediante el entrenamiento continuo y no aparecen aisladas sino integradas en un sistema. El trabajo con las habilidades presupone llevar a la práctica los conocimientos adquiridos y los modos de realización de la actividad.

La función del profesor es de consultante y guía científico educativo lo que potencia el trabajo coparticipativo en equipo, propiciando un clima afectivo de intercambio donde el conocimiento del estudiante aflora y le permite al profesor percibir las potencialidades de cada estudiante y para el tratamiento de sus diferencias individuales en la formación de la habilidad *programar* en las asignaturas de Programación de los primeros años en la carrera de Ingeniería en Sistemas Computacionales.

La solución de las tareas parte de la consideración del aspecto psicológico, en la necesidad de que se reflejen procedimientos racionales de la actividad mental. En el planteamiento de la tarea debe manifestarse la contradicción entre lo conocido y lo desconocido como motor impulsor para su solución.

Cada tarea se determina por los objetivos y su duración en la enseñanza, por el carácter del contenido y por las condiciones materiales en que se realiza. La solución de la tarea implica la transformación del propio sujeto actuante y en algunos casos, la del objeto de estudio. La misma cumple determinadas funciones didácticas dentro del proceso docente educativo. Esta tiene un aspecto intencional (el objetivo) y un aspecto operacional formas y métodos y debe revelar la fusión de la instrucción y la educación.

El proceso de desarrollo de la tarea integradora transcurre por las etapas de motivación, orientación, desarrollo, ejecución y evaluación.

En este trabajo se defiende que un sistema de tareas integradoras contextualizadas en función de los objetivos de la disciplina, que reflejen la integración entre las asignaturas Programación I, II, III y Orientada a Objetos de los primeros años con las restantes del pensum curricular, en la formación de ingenieros en sistemas computacionales, contribuye a la formación profesional y debe ser por tanto, objeto del trabajo metodológico de los profesores su determinación, orientación, desarrollo y evaluación.

El Sistema de Tareas Docentes Integradoras que se fundamenta, se caracteriza por ser estructurado de los contenidos esenciales de las asignaturas de programación de computadoras. Las tareas resultan de cada invariante de habilidad por las asignaturas se caracterizan por la interrelación de los contenidos entre sí, los cuales posibilitarán que el estudiante: dedique más tiempo a la actividad de estudio; aplique los contenidos en la práctica; aprendan los nuevos contenidos; busque independientemente la información necesaria para vencer la contradicción fundamental del proceso; forme, consolide y desarrolle habilidades para la utilización del contenido; regule y autoregule la asimilación de los contenidos; se preparen para nuevas tareas docentes; desarrolle la independencia

cognoscitiva de las asignaturas de programación en los primeros años de la carrera de Ingeniería en Sistemas Computacionales.

El Sistema de Tareas Docentes Integradoras se estructura a través de las acciones y operaciones donde el estudiante va incursionando en los diferentes niveles de aprendizaje (reproductivo y productivo) de las asignaturas de programación de computadoras de los primeros años de la carrera, a partir de una base orientadora para la acción, gestionada por el profesor y con la participación activa y consciente de los estudiantes, imprimiéndole su estilo y ritmo de trabajo atendiendo a sus características personales y a su nivel de aprendizaje individual en la aplicación de las tareas, convirtiéndose el estudiante en el protagonista de su propio aprendizaje.

El Sistema de Tareas Docentes Integradoras se fundamenta en la relación de los componentes del proceso docente educativo para contribuir a la formación de la habilidad *programar*.

El proceso de formación de las habilidades en el ingeniero en sistemas computacionales, precisó de un análisis de los fundamentos teóricos de la didáctica; así como de las leyes que rigen el comportamiento del proceso docente educativo en los diferentes temas de las asignaturas de programación de computadoras.

Se consideraron los objetivos como la categoría rectora del proceso, se formularon y dimensionaron en cada uno de los temas de las asignaturas de programación de computadoras en los primeros años de la carrera, en función de lograr las transformaciones en el aprendizaje de los estudiantes que contribuyen a la formación de la habilidad *programar*, precisando en estos objetivos su función educativa, instructiva y desarrolladora.

Los objetivos educativos, instructivos y desarrolladores se refieren a la formación de convicciones y rasgos de la personalidad, a la asimilación de los conocimientos y a la

formación de habilidades en los estudiantes en cada tema de la asignatura, logrando su cumplimiento mediante la realización de acciones y operaciones que contribuyen a resolver el problema, la formación de la habilidad *programar* incidiendo en el modo de actuación del ingeniero en sistemas computacionales.

Las relaciones internas entre los objetivos, contenidos y métodos (formas y medios) se pusieron de manifiesto en la estrategia al seleccionar y estructurar el contenido de la enseñanza partiendo del sistema de conocimientos, habilidades y valores en las asignaturas de programación de computadoras en los primeros años en correspondencia con los objetivos de cada tema de estas asignaturas.

El sistema de conocimientos, habilidades y valores se seleccionaron, organizaron y estructuraron en cada uno de los temas de la asignatura en función de lograr las transformaciones en el aprendizaje de los estudiantes, mediante el cumplimiento de las acciones y operaciones indicadas en las tareas docentes de tipo problémico con niveles de asimilación del contenido reproductivo, productivo y creativo en los diferentes temas de la asignatura, contribuyendo a la formación de la habilidad *programar* en los Ingenieros en Sistemas.

Los métodos, formas y medios que se emplearon en cada una de las tareas de las asignaturas de Programación I, II, III y Orientada a Objetos se caracterizaron por ser motivantes, comunicativos y educativos contextualizándose en la medida en que se vinculó al estudiante a resolver situaciones problémicas reales o simuladas de la producción de software, mediante la realización de acciones y operaciones cada vez más exigentes por el grado de complejidad de las tareas docentes de tipo problémico, constituyendo esto la verdadera actividad transformadora del estudiante, ya que con el empleo de los métodos, formas y medios no solo convierte el contenido y determina su significación sino que se

apropia de él connotándolo, contribuyendo a la formación de la habilidad *programar* y en el modo de actuación del ingeniero en sistemas computacionales.

La evaluación del cumplimiento de las acciones y operaciones que contribuyen a resolver el problema, la formación de la habilidad *programar* está basado en un alto componente de autoevaluación, de intercambio entre el profesor- estudiante, de muestra de resultados parciales en el seno del grupo realzando el carácter educativo, sistemático y de autorregulación del proceso. El control y evaluación de las acciones y operaciones se realizan en cada tarea docente de tipo integrador, donde el estudiante tiene que exponer ante el profesor y el grupo el trabajo realizado en la situaciones problémicas propuestas o integradoras mediante proyectos, seminarios y talleres con su correspondiente evaluación, contribuyendo a la formación de habilidades de comunicación, trabajo en grupo y a la habilidad *programar* en cada tema de la asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales.

2.3 Concepción del Sistema de Tareas Docentes Integradoras para la formación de la habilidad *programar* en las asignaturas de programación de computadoras en los primeros años en la carrera de Ingeniería en Sistemas Computacionales.

El Sistema de Tareas Docentes Integradoras que se propone en el presente trabajo, estructura los contenidos esenciales de las asignaturas: Programación I, II, III y Orientada a Objetos que han de contribuir a la solución de situaciones problémicas, mediante la construcción de programas para computadoras, en las que contribuyen los conocimientos procedentes de las asignaturas de programación que pertenecen a los primeros años del currículo. En este sentido los contenidos esenciales están insertados en un marco más amplio que permite integrarlos en niveles superiores de generalización a las que ofrece una sola asignatura de programación.

El proceso docente debe desarrollarse a partir del planteamiento de un problema por parte del profesor, cuya solución significa la asimilación del estudiante de un nuevo contenido por medio del cual se resuelve la situación problémica. El objetivo estará dirigido al vencimiento del problema propuesto. La tarea será el modo en que se concreta el objetivo, en que se resuelve el problema por programación. La habilidad *programar* que dominará el estudiante será la potencialidad que quedará en él después de vencida la tarea, además de la apropiación del contenido

En este sistema de tareas se manifiestan las relaciones internas entre los objetivos, contenidos y métodos al seleccionar y estructurar el contenido de la enseñanza partiendo del sistema de conocimientos, habilidades y valores en las asignaturas de programación de los primeros años en correspondencia con los objetivos de la asignatura. Cada tarea del sistema establece las acciones y operaciones necesarias para su solución y lo que el estudiante debe desarrollar para integrar el contenido de las asignaturas que pertenecen al área de programación y de esta forma ayudar a la preparación del estudiante para su futuro quehacer laboral.

El Sistema de Tareas Docentes Integradoras que se plantea consta de tres etapas fundamentales: Orientación, Ejecución y Evaluación que desarrolladas armónicamente permiten llevar a cabo el sistema de acciones y operaciones en forma de tareas docentes de tipo integrador para que el estudiante en las asignaturas de programación de computadoras de los primeros años de la carrera de Ingeniería en Sistemas Computacionales, alcance a formar la habilidad *programar* que consiste en *concebir el problema* (análisis), *diseñar el algoritmo de solución* (trazar el plan), *traducir el algoritmo a un lenguaje de programación* (ejecutar el plan) y *depurar el programa* (revisar el plan) que soluciona el problema, estos

cuatros factores considerados como las invariantes en el proceso de formación y desarrollo de dicha habilidad.(Anexo No. 04)

2.3.1. Etapa de Orientación.

El momento didáctico correspondiente a la Orientación se lleva a cabo fundamentalmente en la primera asignatura de programación: Programación I; en donde debería iniciarse la formación de la habilidad *programar*, en toda su integridad**.** El estudiante recibe la orientación en la semana inicial del periodo lectivo de la asignatura a través de una actividad de dos horas de duración que tiene en la conferencia su forma organizativa docente.

La conferencia orientadora inicial tiene como propósitos en primer lugar, lograr una disposición positiva del estudiante para utilizar activa, consciente y sistemáticamente el sistema de tareas docentes; así como dotarlo de la representación mental de las ejecuciones a llevar a cabo y de los nuevos conceptos necesarios para ello.

La disposición positiva del estudiante se consigue haciéndole comprender que el conocimiento y las habilidades que ya posee, relacionadas fundamentalmente con la concepción y análisis del problema a programar, no son suficientes para llevar a cabo su labor profesional, creándole una contradicción entre lo que sabe y es capaz de hacer por un lado, y lo que necesita saber y ser capaz de hacer, por otro. La comprensión por el estudiante del carácter esencial de la habilidad *programar* constituye el elemento clave para transformar la necesidad de aprendizaje en motivación, esta última, base de cualquier actividad.

El estudiante recibe la Base Orientadora de la Acción BOA en correspondencia con el elevado grado de dificultad de la habilidad a formar y además, teniendo en cuenta como punto de partida el nivel de formación propedéutico de la habilidad, pues el estudiante de

los primeros años de la carrera de Ingeniería en Sistemas en la actualidad cuenta con presupuestos insuficientes.

Esta situación determina que el profesor explique detalladamente los nuevos conceptos, las nuevas definiciones que se introducen y cada una de las acciones a llevar a cabo por el estudiante para el aprendizaje de la habilidad, fundamente la importancia de dichas ejecuciones y ejemplifique con situaciones de la práctica que pudieran ser ya familiares para algunos estudiantes, todo lo cual facilita la comprensión de las mismas, caso contrario la línea de partida será desde cero.

En la conferencia orientadora inicial el estudiante recibe aquellas indicaciones que le permitirán la elaboración de un programa mental, a manera de representación interna de la habilidad, que le posibilite a su vez la autoevaluación de su progreso, el autocontrol y la regulación de su aprendizaje. Por todo ello, con la orientación inicial el estudiante sabe qué hacer y cómo hacerlo, conoce el por qué debe aprender la nueva habilidad, y cómo puede controlar por sí mismo su aprendizaje.

La orientación continúa en cada una de las áreas y asignaturas de la profesión, y a nivel de estas en cada uno de los temas, donde el estudiante recibe conferencias que introducen nuevos contenidos. Los nuevos contenidos que el estudiante incorpora en estas conferencias, y también mediante el estudio independiente, van conformando el cuerpo de conocimientos o base gnoseológica de la habilidad *programar.*

El Sistema de Tareas docentes propuesto, en esta etapa emplea varios tipos de Base Orientadora para la Acción BOA atendiendo a sus características según su carácter generalizado, su plenitud y su modo de obtención; basado en la utilización de los programas de computadoras en Problemas Resueltos, Propuestos e Integradores.

En los Problemas Resueltos la BOA por su carácter generalizado es concreta, por su plenitud es completa y por su modo de obtención se da preparada con intencionalidad. En los Problemas Propuestos e Integradores por su carácter es generalizada, por su plenitud es completa y por su modo de obtención es elaborada independientemente.

En la propuesta se aprecia una combinación de sus tipos en los Problemas Resueltos la BOA es de Tipo II con la característica de ser concreta por el carácter generalizado, es completa por su plenitud y se da preparada por su modo de obtención.

En los Problemas Propuestos e Integradores es de Tipo III ya que por su carácter es generalizada por su plenitud es completa y por el modo de obtención es elaborada independientemente.

Esta etapa se caracteriza por ofrecer la BOA al estudiante, desarrollándose los eslabones de motivación y comprensión del contenido. En esta primera etapa en correspondencia con los objetivos se le indica al estudiante las primeras acciones a realizar dirigidas a la apropiación de conceptos, leyes lógicas, definiciones, principios, técnicas mediante situaciones problémicas utilizando vías tales como: descomposición de problemas, identificación de los datos de entrada, auxiliares, de salida, diseño de programas en diagramas de flujo, diseño de programas en lenguaje pseudocódigo, diagramas del Modelaje Unificado de Objetos UML, Estructuras de Control Estáticas como cadenas, arreglos y Dinámicas como Listas, Pilas, Colas, Árboles y Grafos, para clases y objetos las propiedades de encapsulamiento, herencia, polimorfismo, reusabilidad, en estas vías se intensifica el interés hacia lo conocido, constituyendo una premisa para el desarrollo de discusiones heurísticas del material docente en las que el profesor conduce al desarrollo del pensamiento crítico mediante las reflexiones de los estudiantes y se propicia un clima afectivo de intercambio donde el conocimiento experiencial del estudiante aflora y propicia al docente las

potencialidades para el tratamiento de las diferencias individuales. Las acciones en esta etapa están a un nivel de asimilación del conocimiento de forma reproductiva lleva implícito un proceso de familiarización, exige que el estudiante sea capaz de repetir el contenido que se le ha informado ya sea en forma declarativa o resolviendo problemas similares a los resueltos, con un grado de complejidad acorde a este nivel, es decir, las acciones y operaciones que el estudiante realiza en esta etapa son observar, describir, comparar, caracterizar, identificar y analizar.

En esta etapa de orientación el estudiante realiza estas acciones y podrá estudiar situaciones reales o simuladas de la programación de computadoras a través de Problemas Resueltos que orientan al estudiante a cómo enfrentar las situaciones problémicas, estas serán explicadas en clase enfatizando en el proceso de enseñanza los procedimientos racionales para la solución de problemas siguiendo los fundamentos de programación presentado en las conferencias, clases de ejercitación y clases prácticas dictadas. En el Nivel Reproductivo de asimilación del conocimiento, el estudiante comienza a familiarizarse con el método de solución de problemas mediante la programación de computadoras y aplicando en forma gradual y consciente las habilidades en formación, haciendo especial énfasis en la formación de la habilidad *programar* ya que las situaciones problémicas presentadas en los talleres y laboratorios de computación, están dirigidas a la concepción y análisis del problema, el diseño del algoritmo solución ya que estos factores constituyen la programación y es precisamente lo que debe dominar el estudiante.

En los laboratorios de programación de computadoras se puede apreciar cómo al estudiante se le orientan las acciones y él por sí solo dará respuesta a interrogantes que llevan implícitos el conocer y saber los conocimientos recibidos al nivel de asimilación reproductivo.

La BOA está presente en cada tarea docente de tipo integrador, atendiendo al nivel de asimilación del conocimiento reproductivo, productivo y creativo a través de las acciones y operaciones seleccionadas y organizadas de acuerdo con el nivel de complejidad; esto se aprecia en la acciones y operaciones que el estudiante tiene que realizar al enfrentarse a situaciones problémicas que le presentan los Problemas Propuestos en los que ya tiene que accionar y operar llegando explicar, valorar, argumentar, plantear estando estas en un nivel de asimilación del conocimiento de tipo productivo y en los Problemas Integradores en los que ya tiene que realizar acciones y operaciones del orden de determinar, generalizar, demostrar, diseñar y aplicar, que es el nivel más alto de lo productivo, el estudiante tiene que hacer aportes novedosos para él, utilizando la lógica de la investigación científica.

En esta etapa de orientación se le brinda al estudiante desde la clase, los talleres, proyectos y laboratorios de programación toda la información necesaria pudiéndose observar el incremento en la complejidad de las situaciones problémicas reales o simuladas de la programación de computadoras y la exigencia cada vez mayor del cumplimiento de las acciones y operaciones a las que tiene que enfrentarse el estudiante para lograr dominar la habilidad *programar*.

2.3.2 Etapa de Ejecución.

Este momento de Ejecución del Sistema de Tareas Docentes se caracteriza por exigir al estudiante que aplique los conocimientos de programación ante nuevas situaciones problémicas con un nivel de asimilación productivo y un mayor grado de complejidad. El estudiante ya tiene que accionar y operar llegando a ejemplificar, explicar, valorar, clasificar, argumentar, relacionar e interpretar, estando estas a un nivel de saber programar, propios de los Problemas de Programación Propuestos en las cuatro tareas de las asignaturas de programación en los primeros años de la carrera de Ingeniería en Sistemas,

que indudablemente tienen un grado de complejidad superior a los ya analizados en la etapa de orientación. En esta etapa se le presentan situaciones problémicas con carácter individual, es decir, un problema a resolver por programación distinto para cada estudiante en la que a través de las acciones y operaciones seleccionadas y organizadas le imprimirá su propio ritmo de trabajo.

Es característico en esta etapa que el papel del profesor es de consultante y guía científico por lo que a través de seminarios, talleres, sesiones de laboratorios de programación, visitas a empresas de desarrollo de software, apoyan el trabajo individual de los estudiantes con suficiente flexibilidad para que manejen su propio estilo de programación.

Las situaciones problémicas propuestas reales o simuladas de la programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales, ilustran al estudiante mediante problemas que involucran diagramas de flujo, lenguaje pseudocódigo, líneas de comandos de un lenguaje de programación, pruebas de escritorio y demás esquemas de modelaje de clases y objetos UML, para que los estudiantes observen las fases de construcción de un programa de computadoras visto desde la concepción del problema, el diseño del algoritmo, su traducción al lenguaje de programación y la fase de depuración del programa emergiendo estos factores como invariantes en el proceso de formación de la habilidad *programar*, esto posibilita que el estudiante interiorice estas facetas a tener en cuenta para *programar* constituyendo la habilidad profesional del ingeniero en sistemas computacionales y que se vaya apropiando del método de solución de problemas por programación de computadoras.

El nivel de asimilación de los conocimientos y el grado de complejidad de las situaciones problémicas se incrementan al presentar las Problemas Integradores, estos tienen la característica de estar a un nivel creativo que es el nivel más alto de lo productivo en que el

estudiante tiene que hacer aportes novedosos para él, utilizando la lógica de la programación y de la investigación científica, puede apreciarse que las acciones y operaciones tiene un nivel de determinar, generalizar, demostrar, realizar, aplicar, diseñar, analizar, sintetizar e identificar; indudablemente tiene un grado de complejidad superior a las presentadas en los Problemas Resueltos y Propuestos.

La situaciones problémicas reales o simuladas de la programación de computadoras presentadas en los Problemas Integradores han sido objeto de discusión y se concluye que logran que el estudiante se motive, al dar respuesta a una situación problémica real o simulada; pero con un nivel de complejidad superior respecto a la orientación y reproducción, es lógico suponer que los Problemas Integradores presentados en los temas finales de las asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales, tendrán un mayor nivel de complejidad ya que integran los conocimientos ya adquiridos a lo largo de los cursos.

En la programación de estos Problemas Integradores el estudiante es capaz de realizar la concepción del problema mediante su *análisis* exhaustivo, el *diseño del algoritmo* solución, la codificación o *traducción del algoritmo* al lenguaje de programación seleccionado y la *depuración final del programa* de computadora que soluciona al problema, resultando estos factores, las invariantes para el dominio de la habilidad *programar*. (Anexo No. 05)

El nivel creativo de asimilación de la habilidad *programar* es muy difícil de lograr en el proceso docente educativo de los primeros años de la carrera de Ingeniería en Sistemas Computacionales. Para alcanzar dicho nivel, el estudiante tendría que, una vez adquirida la habilidad al nivel productivo, aplicar el sistema de tareas en la solución de problemas de programación no necesariamente contenidos en los temas de la asignatura o en problemas de evidente complejidad, y cuya solución le exija no solo la indagación, la búsqueda del

conocimiento necesario para ello, sino también, la utilización de un pensamiento creativo; cuestión que es poco probable dada la inexperiencia del estudiante y la ausencia de una total independencia de este en el aprendizaje particular de esta habilidad. No obstante, algunos estudiantes revelan un potencial en las habilidades del pensamiento lógico y son claras posiciones antagónicas de esta afirmación.

El estudiante trabaja con el sistema de tareas docentes para la formación de la habilidad *programar* utilizando los conocimientos asimilados según la secuencia en que los contenidos están distribuidos en el programa de estudio de cada asignatura de programación. Este aprendizaje planificado se combina con el aprendizaje incidental propio de la enseñanza en las condiciones reales de la práctica de programación, concreción a su vez en la educación de uno de los principios didácticos: la vinculación del estudio con el trabajo. El aprendizaje incidental implica la ejercitación del estudiante en problemas que resolver por programación no necesariamente enmarcados en el tema correspondiente al programa de estudio por la cual transita.

La realización de las acciones por el estudiante, del sistema de tareas docentes de forma incidental o no planificada, está condicionada por la velocidad con que el estudiante va dominando las distintas ejecuciones que componen la habilidad *programar*. Constituye además, una vía para que el profesor regule la carga y el ritmo de trabajo de cada estudiante; así como para determinar el nivel de asimilación que cada uno de ellos puede proponerse en un momento dado, individualizando con ello el proceso de formación de la habilidad.

Es necesario destacar que para la formación de la habilidad *programar* el estudiante va a ejecutar tareas de familiarización, reproducción, producción o creación, en el ritmo con que los estudiantes dominan las ejecuciones de la habilidad, lo que determina la manera de

conducir el proceso docente de forma tal que, al aumentar el grado de dificultad de los problemas a programar, le permita al grupo en general y a los estudiantes en particular, el paso de un nivel de asimilación del contenido a otro, siempre considerando en este proceso las variaciones individuales en la progresión determinadas por las diferencias entre los estudiantes. Por esta razón, no existe un lapso de tiempo prefijado para la formación de la habilidad en un nivel de asimilación específico; ni todos los estudiantes van alcanzando cada nivel de asimilación de forma simultánea y uniforme.

No obstante el sistema estructurado propiciará la incursión de los estudiantes en forma gradual contribuyendo a las diferentes formas en función de los métodos que exigen cada tarea y el tiempo que tenga el programa.

2.3.3 Etapa de Evaluación.

El momento correspondiente a la Evaluación como medida para valorar el cumplimiento de los objetivos serán orientados a controlar el aprendizaje del estudiante mediante el Sistema de Tareas Docentes Integradoras para la formación de la habilidad *programar*, se lleva a cabo sistemáticamente de forma armónica e integrada.

Según la ya expuesta dinámica de participación del profesor y del estudiante en dicha ejecución, los momentos iniciales exigen una intensa actividad de control por parte del profesor con vistas a garantizar las correcciones necesarias para el dominio del sistema operacional de la habilidad, de forma tal que la ejercitación posterior cada vez más independiente garantice su correcta formación.

En la medida que se va formando la habilidad *programar* y que el estudiante logra conformar un proceso mental para evaluar su desempeño, adquiere progresivamente una gran relevancia la autoevaluación, como vía para la autorregulación del aprendizaje.

La evaluación frecuente se lleva a cabo fundamentalmente en las actividades propias de la educación en el laboratorio de programación durante la interacción estudiante – programa de computadoras: concepción y análisis del problema, diseño del algoritmo, codificación y depuración del programa (Anexo No. 04), también en actividades de carácter académico como los proyectos, clases prácticas: análisis de problemas resueltos, resolución de problemas propuestos e integradores, seminarios, laboratorios de programación, mediante la utilización de situaciones simuladas.

La evaluación sistémica permite constatar el progreso del grado de independencia y dominio que va logrando el estudiante en la ejecución de tareas y de la habilidad *programar.*

Para la evaluación final se utiliza el proyecto integrador al término de las asignaturas en los primeros años de carrera, en el cual la habilidad *programar* es aplicada a la programación de un problema real que integre las asignaturas a nivel semestral.

La complejidad de la formación de la habilidad *programar* determina una permanente elevada participación del profesor en la evaluación y control del progreso en la adquisición de la habilidad.

Esta etapa en el Sistema de Tareas Docentes Integradoras se basa en un alto componente de autoevaluación, de intercambio entre profesor- estudiante, de mostrar los resultados parciales en el curso, que culmina con una exposición – ante sus compañeros y el profesor- de los resultados obtenidos. Las acciones y operaciones de evaluación realzan el carácter educativo y de autorregulación del proceso ya que en la medida que el estudiante se retroalimenta, permite modularlo.

Las acciones y operaciones de Evaluación que se realizan en cada tarea docente de tipo integrador son sistemáticas, ya que el estudiante individualmente tiene que exponer ante el

profesor y el grupo, el trabajo realizado, la solución de las situaciones problémicas propuestas o integradoras mediante programas de computadoras construidos por él, en las formas organización de seminarios, talleres, laboratorios; en el caso de los Problemas Propuestos cada estudiante tendrá que enfrentar una situación problémica diferente y defender los resultados obtenidos, estas acciones y operaciones las realiza de forma controlada por el profesor, quien podrá ir evaluando el cumplimiento de las acciones y operaciones en el proceso de formación y desarrollo de la habilidad *programar*.

La evaluación que realiza el profesor del cumplimiento de las acciones y operaciones seleccionadas, organizadas y presentadas en los Problemas Propuestos e Integradores es sistemática ya que a través de un registro académico diario, el profesor llevará un control de las actividades efectuadas por los estudiantes.

Por la complejidad de las situaciones problémicas y la forma sistemática en que se realiza la evaluación de las acciones y operaciones, en el caso de que algún estudiante no cumpla las tareas, es decir, no demuestre dominio en su trabajo y disertación, entonces después de haber aclarado las dudas con el profesor, se le planteará en clase otra situación problémica con un grado de dificultad similar.

2.4 Planteamiento del Sistema de Tareas Docentes Integradoras para formar la habilidad *programar* en los asignaturas de programación de computadoras de primeros años en la carrera de Ingeniería en Sistemas Computacionales.

La tarea docente es la célula básica del proceso docente educativo se fundamenta en la acción tanto del profesor como de los estudiantes dentro del proceso, con el fin de alcanzar los objetivos propuestos. La concepción de estructurar las tareas docentes integradoras en forma sistemática, brinda la posibilidad de lograr las transformaciones en el aprendizaje de

los estudiantes de las asignaturas de programación de computadoras en los primeros años de la carrera y contribuyen a la formación de la habilidad *programar*.

Se concibió estructurar tareas docentes integradoras de tipo reproductivo, ya que estas propician que el estudiante sea capaz de repetir el contenido que se le ha informado en las conferencias, ya sea en forma declarativa o resolviendo problemas similares a los ya resueltos con asistencia del profesor, las tareas de tipo productivo posibilitan que el estudiante aplique los conocimientos y habilidades de programación ante nuevas situaciones y resuelva problemas de la programación estructurada y orientada a objetos, con diferentes niveles de complejidad y en las de tipo creativo el estudiante demuestre aportes novedosos de la técnica, utilizando la lógica de la programación, al escribir programas de computadoras inéditos como vías de solución creativa, para problemas reales o simulados de desarrollo de software con mayor nivel de complejidad.

Se decide emplear las tareas docentes de tipo integrador, estructuradas a partir de las invariantes en los contenidos de las asignaturas de Programación I, II, III y Orientada a Objetos del pensum curricular planificado para los primeros años de la carrera de Ingeniería en Sistemas Computacionales (Anexo No. 01); (Anexo No. 02). El desarrollo de las tareas se basa en la sistematización de los temas esenciales de la programación de computadoras desde el paradigma estructurado hasta el enfoque actual de orientación a objetos, lo que posibilitará al estudiante construir programas desde la reproducción hasta el nivel productivo y creativo, apropiándose de las técnicas de programación como una vía de solución de problemas, contribuyendo mediante la realización de acciones y operaciones a la formación de la habilidad *programar* en el ingeniero en sistemas computacionales.

El sistema de tareas docentes integradoras elaborado incluye las categorías didácticas, presididas por un objetivo trascendental centrado en una habilidad de aplicación a lograr,

esto justifica la necesidad de la integración del contenido de las asignaturas de programación de los primeros años, que irá acercando gradualmente al estudiante a formar la habilidad *programar* en la ejecución de cada una de las tareas planificadas.

La propuesta tiene una estructura organizativa que permite abordar las asignaturas de programación de computadoras en las Etapas de Orientación, Ejecución *reproductiva, productiva y creativa*, y la Etapa de Evaluación; mediante el cumplimiento de las acciones y operaciones seleccionadas y organizadas en la estructura del sistema, que exigen un alto componente de autopreparación en los estudiantes, que los llevará al dominio de los conocimientos, de las habilidades lógicas y en especial a la formación de las habilidades para concebir el problema, diseñar el algoritmo de solución, traducirlo a un lenguaje de programación y depurarlo, las que constituyen las invariantes funcionales de la habilidad *programar.* (Anexo No. 04)

Se expone la organización del sistema de tareas integradoras en las cuatro invariantes de las asignaturas de programación de computadoras de los primeros años de la carrera, para cada una de las etapas que conforman el sistema propuesto.

Tarea Docente Integradora I: *Concebir el problema a programar*.

Se expone el objetivo a lograr en la tarea: *Analizar el dominio contextual del problema e identificar los datos de entrada, datos de salida, datos auxiliares, objetos, clases, atributos, condiciones, restricciones y la descripción de la solución del problema, a resolverse por programación.*

Este objetivo está formulado y dimensionado en función de lograr las transformaciones en el aprendizaje de los estudiantes mediante las acciones y operaciones en las diferentes etapas del sistema.

Etapa de Orientación de la Tarea Docente Integradora I.

Esta etapa se caracteriza por ofrecer al estudiante la Base Orientadora para la Acción BOA en correspondencia con el objetivo del tema, se le indica al estudiante las primeras acciones a realizar, dirigidas a la apropiación de conceptos, principios y definiciones, mediante la construcción de cuadros sinópticos y mapas conceptuales de los elementos teóricos más esenciales, el estudio y descomposición de enunciados de problemas elementales, identificación de los datos de entrada, auxiliares, y de salida, las clases y sus objetos, utilizando diferentes vías: revisión de ejercicios resueltos del texto guía, y en referencias bibliográficas complementarias, observar y comentar problemas resueltos de fuentes validadas por el profesor provenientes de la internet. Los estudiantes apoyados en estas vías, intensifican el interés hacia lo conocido, esto constituye una premisa básica para lograr la motivación y comunicación con el estudiante. Las acciones en esta etapa están al nivel de asimilación reproductiva, llevan implícito un proceso de familiarización, exigen del estudiante el repetir el contenido que se le ha informado en forma declarativa o analizando problemas sencillos, es decir, las acciones y operaciones que el estudiante realiza en esta tarea docente integradora, están al nivel de observar, describir, comparar, descomponer, caracterizar, identificar, analizar.

En esta etapa de orientación, mediante la revisión de problemas resueltos, el estudiante podrá enfrentar la concepción y análisis de los enunciados de estos problemas, y la identificación de los datos de entrada, auxiliares y de salida; estas acciones serán explicadas en conferencias y clases prácticas. En este nivel de asimilación del conocimiento reproductivo, el estudiante empieza a familiarizarse con el primer paso del método de solución de problemas mediante programación de computadoras, que se constituye en la

primera invariante funcional *Concebir el problema* en la formación de la habilidad *programar*.

En el Sistema de Tareas Integradoras (Anexo No. 05) se pueden apreciar las orientaciones brindadas por el profesor en la revisión de problemas resueltos; el estudiante en su tiempo de auto preparación accederá a las acciones y operaciones correspondientes y tendrá la posibilidad de *observar, descomponer, comparar*, *identificar* sin la presencia del profesor; pero con las orientaciones precisas para la asimilación de los conocimientos a este nivel reproductivo.

Etapa de Ejecución de la Tarea Docente Integradora I.

Esta etapa se caracteriza por exigir al estudiante la aplicación ante las tareas docentes integradoras de tipo problémico acorde a un nivel productivo de asimilación del conocimiento, la realización de acciones y operaciones propias de los ejercicios propuestos de la Tarea I (Anexo No. 05)

Las situaciones problémicas reales o simuladas del análisis de problemas tienen un carácter individual, es decir, una variante de solución diferente para cada estudiante. El desempeño del profesor es de consultante y guía educativo científico, ya que mediante el desarrollo de las clases prácticas, sesiones de laboratorios de programación permite al estudiante darle cumplimiento a las acciones y operaciones indicadas.

El nivel de asimilación del conocimiento y el grado de complejidad de las tareas docentes integradoras de tipo problémico, se incrementan al presentar Ejercicios Integradores; estos tienen las características de un Nivel Creativo, la realización de las acciones y operaciones para el Problema Integrador de la Tarea I (Anexo No. 05)

Etapa de Evaluación de la Tarea Docente Integradora I.

Esta etapa se caracteriza por mantener un control y evaluación de la ejecución de las acciones y operaciones realizadas por los estudiantes, que se contextualiza en la participación del estudiante en las actividades docentes conferencias, clases de ejercitación grupal e individual, clases prácticas y talleres.

El control y la evaluación realzan el carácter educativo y de autorregulación del proceso ya que en la medida en que el estudiante se retroalimenta, permite modularlo.

Las situaciones problémicas propuestas e integradoras del tema se controlan y evalúan mediante la participación de los estudiantes en las clases prácticas, seminarios, talleres, en el caso de los Problemas Propuestos, cada estudiante tendrá que enfrentar una situación problémica diferente, y defender su análisis.

El control y evaluación que realiza el profesor del cumplimiento de las acciones y operaciones es sistemático, el profesor registra la participación del estudiantes basada en la calidad de respuestas a las preguntas efectuadas por el profesor, su nivel de actuación en clase, esto es un indicador que el profesor utiliza como mecanismo de control de las acciones y operaciones efectuadas por los estudiantes.

En esta etapa se evalúan las transformaciones en el aprendizaje de los estudiantes y su contribución a la formación de la habilidad *programar*. (Anexo No. 05)

Tarea Docente Integradora II: *Diseñar la solución del problema a programar.*

Se expone el objetivo a lograr en la tarea: *Definir la estrategia de solución del problema a programar mediante el diseño de la lógica basada en algoritmos y modelos de la solución, desde los paradigmas de la programación estructurada y orientada a objetos*

Este objetivo se formula y dimensiona en función de alcanzar las transformaciones en el aprendizaje de los estudiantes mediante el cumplimiento de las acciones y operaciones en las diferentes etapas del sistema propuesto.

Etapa de Orientación de la Tarea Docente Integradora II.

Esta etapa se caracteriza por brindar al estudiante la Base Orientadora para la Acción BOA, en correspondencia con el objetivo de la tarea docente integradora, se le indica a los estudiantes las primeras acciones dirigidas a la apropiación de conceptos, leyes lógicas, principios, definiciones mediante el diseño de situaciones problémicas acorde con el nivel de asimilación reproductivo del conocimiento, utilizando diferentes vías: diseño de algoritmos estructurados basados en diagramas de flujo, diseño estructurado basado en pseudocódigo, diseño de clases y objetos basados en los trece tipos de diagramas estandarizados UML (Anexo No. 05). Las acciones en esta etapa están a un Nivel Reproductiva de asimilación del conocimiento, llevando implícito un proceso de familiarización, exigen del estudiante el reproducir el contenido de conceptos, leyes lógicas, principios y estándares de diseño, que posteriormente aplicarán en la tarea y la resolución de situaciones problémicas con un grado de complejidad acorde con este nivel de asimilación del conocimiento, es decir, las acciones y operaciones que el estudiante realiza en las tareas docentes de tipo problémico están al nivel de estudiar las diferencias de diseños con enfoque estructurado y con orientación a objetos, observar y comparar las diferentes tipos de representación del diseño, gráfico o escrito en lenguaje pseudocódigo y sus características, además de identificar los atributos de modelación de clases y objetos del problema y la descripción de las condiciones suficientes y necesarias para su diseño.

En esta etapa el estudiante podrá observar, comparar las situaciones reales o simuladas de diseño de algoritmos presentadas en el (Anexo No. 05), mediante los Diseños Resueltos,

estos serán explicados por el profesor siguiendo la metodología orientada en las conferencias, seminarios, clases prácticas que constituyen las formas organizativas para la formación de la habilidad *programar*.

Etapa de Ejecución de la Tarea Docente Integradora II.

Esta etapa se caracteriza por exigir al estudiante la aplicación ante las tareas docentes integradoras de tipo problémico a un nivel de asimilación del conocimiento productivo, la realización de acciones y operaciones propias de los Problemas Propuestos de la Tarea II Diseñar la solución del problema a programar (Anexo No. 05)

El nivel de asimilación del conocimiento y el grado de complejidad de las tareas docentes de tipo problémico se incrementan al presentar los Problemas Integradores; estos tienen las características de estar a un nivel creativo, la realización de las acciones y operaciones para el Problema Integrador de la Tarea II (Anexo No. 05)

Etapa de Evaluación de la Tarea Docente Integradora II.

Las situaciones problémicas propuestas e integradoras del tema se evalúan mediante la participación de los estudiantes en clases prácticas, seminarios, talleres, sesiones de laboratorios de programación en las que el estudiante defiende la realización de las acciones y operaciones ejecutadas individualmente ante el profesor y el grupo. Está presente también el registro de control de la participación activa de los estudiantes.

Tarea Docente Integradora III: *Traducir a un lenguaje de programación, la solución del problema a programar.*

El objetivo a lograr en la tarea docente: *Codificar el algoritmo y/o el modelo orientado a objetos (UML) mediante un lenguaje de programación para la solución del problema a programar.*

Se presenta esta tarea al estudiante en función de lograr las transformaciones en el aprendizaje de programación de los estudiantes mediante la realización de la propuesta en todas sus etapas didácticas.

Etapa de Orientación de la Tarea Docente Integradora III.

En este tema la etapa se caracteriza por brindar a los estudiantes la Base Orientadora para la Acción BOA, en correspondencia con el objetivo de la tarea, indicándole las acciones mediante experiencias de cátedra, el dominio de lenguajes de programación de mayor uso en el mercado, mediante estas vías los estudiantes se apropian de los comandos, estructura de sintaxis del lenguaje de programación, técnicas de programación, estándares, leyes lógicas, definiciones propias de tareas docentes integradoras de tipo problémico al nivel reproductivo, en las que predominarán las acciones al nivel de observar, identificar, describir, codificar las instrucciones del programa de computadora previamente diseñado (Anexo No. 05).

Los Problemas Resueltos por programación presentados por el profesor, serán analizados en clase por el estudiante, asistido por el profesor, basándose en el seguimiento de las líneas de código de diferentes programas y luego explicados en las clases prácticas y laboratorios de programación, impartidas sobre la Tarea.

Etapa de Ejecución de la Tarea Docente Integradora III.

Esta etapa se caracteriza por exigir al estudiante la aplicación de las tareas docentes integradoras de Nivel Productivo de asimilación del conocimiento, la realización de acciones y operaciones mediante la programación de la solución a los Ejercicios Propuestos en la Tarea Integradora III (Anexo No. 05).

El nivel de asimilación del conocimiento y el grado de complejidad de las tareas docentes integradoras de tipo problémico, se incrementan progresivamente, al presentar los

Problemas Integradores que se caracterizan por un Nivel Creativo, la realización de las acciones y operaciones para el Problema Integrador de la Tarea III (Anexo No. 05).

El Proyecto Integrador presentado en este tema, responde a la contradicción planteada en la Etapa de Orientación en la que el estudiante realiza las acciones y operaciones para dar cumplimiento a la tarea docente integradora a un Nivel Creativo, teniendo que determinar, generalizar, aplicar, crear, innovar soluciones informáticas basadas en programación tratados en las Tareas Docentes Integradoras I y II, observándose la deducción, inducción, análisis, síntesis y aplicación de los conocimientos y habilidades ya adquiridos en las tareas precedentes. La situación problémica presentada en el Laboratorio de Programación, posee un grado de complejidad superior a los Problemas Resueltos y Propuestos analizados anteriormente.

En este tema se puede apreciar cómo el estudiante es capaz de darle solución a una situación problémica real o simulada aplicando los conocimientos y habilidades de las Tareas Docentes Integradoras I, II y III lo que demuestra la formación en los estudiantes de la habilidad *programar* incidiendo en el modo de actuación del ingeniero en sistemas computacionales. (Anexo No. 05)

Etapa de Evaluación de la Tarea Docente Integradora de la Tarea III.

La realización de las acciones y operaciones ejecutadas individualmente por los estudiantes se controlan y evalúan durante el desarrollo de la tarea en las clases prácticas, seminarios, talleres, laboratorios de programación, en las que está presente la declaración de datos y variables, la aplicación de la leyes lógicas y utilización de la línea de comandos y uso de librerías en sentencias y estructuras de datos; demostrados por los estudiantes en la defensa realizada ante el profesor y el grupo del cumplimiento de las acciones y operaciones orientadas a lo largo de la tarea.

Tarea Docente Integradora IV: *Depurar el programa.*

Al igual que en las tareas anteriores se expone el objetivo: *Perfeccionar el programa escrito con un lenguaje de programación a fin de que esté listo para la puesta en marcha.* Este objetivo está formulado y dimensionado en función de lograr las transformaciones en el aprendizaje de los estudiantes mediante el cumplimiento de las acciones y operaciones en las diferentes Etapas de Orientación, Ejecución y de Evaluación. (Anexo No. 05)

Etapa de Orientación de la Tarea Docente Integradora IV.

La Base Orientadora para la Acción BOA está en correspondencia con el objetivo de la tarea, se le indica al estudiante las acciones a realizar dirigidas a la apropiación de conceptos, leyes y principios lógicos, definiciones, estándar de revisión, relacionadas con la acciones y operaciones, mediante ejemplos de programas recogidos en clases prácticas y laboratorios, donde se aprecia la importancia de la asimilación reproductiva de los conocimientos de este tema, es decir, que se le informa al estudiante ya sea resolviendo problemas con un grado de complejidad acorde al nivel de observar, identificar, localizar, comparar, mediante la presentación de los Problemas Resueltos que aparecen (Anexo No. 05), que serán explicados con clases de ejercitación.

Etapa de Ejecución de la Tarea Docente Integradora IV.

Esta etapa se caracteriza por exigir al estudiante la aplicación, ante tareas docentes integradoras de tipo problémicas a un Nivel Productivo de asimilación del conocimiento, la realización de acciones y operaciones al nivel de ejemplificar, explicar, valorar, clasificar, argumentar, relacionar, interpretar, desarrollar habilidades lógicas de orden superior en los Problemas Propuestos presentados en (Anexo No. 05).

Las situaciones problémicas reales o simuladas de la programación, tienen un carácter individual, es decir, una variante diferente para cada estudiante. El desempeño del profesor

es de consultante y guía científico ya que mediante el desarrollo de las clases prácticas, seminarios, talleres, sesiones de laboratorios, visitas a entidades conduce el trabajo individual de los estudiantes con suficiente flexibilidad, imprimiéndole su propio estilo de programación y permite al estudiante darle cumplimiento a las acciones y operaciones indicadas en (Anexo No. 05), correspondiente al Sistema de Tareas Docentes Integradoras en estudio.

El nivel de asimilación del conocimiento y el grado de complejidad de las tareas docentes integradoras de tipo problémico se incrementan al presentar el Proyecto Integrador.

La situación problémica que se presenta a los estudiantes en este tema ha sido objeto de discusión en diferentes áreas del currículo del Ingeniero en Sistemas; en las que se le orienta y explica las acciones y operaciones a realizar; así como el cronograma de las actividades prácticas (Anexo No. 05). La ejecución de las acciones y operaciones se realizan en las clases prácticas, seminarios, talleres, sesiones de laboratorios de programación en los que el estudiante expone ante el profesor y el grupo el cumplimiento de las mismas; esta tarea docente de tipo problémico y con un Nivel Creativo de asimilación del conocimiento logra que el estudiante se motive, a medida que él observe que sí puede construir un programa para darle solución a la situación problémica indicada y de que es capaz de realizar las tareas docentes I, II, III y IV resultando las invariantes para la formación de la habilidad *programar* incidiendo en el modo de actuación del ingeniero en sistemas computacionales.

Etapa de Evaluación de la Tarea Docente Integradora IV.

Esta etapa se caracteriza por evaluar la ejecución de las acciones y operaciones realizadas por los estudiantes, que se contextualiza en la participación del estudiante en las actividades docentes conferencias, clases de ejercitación, clases prácticas, seminarios, talleres, sesiones

de laboratorios de programación, visitas en las que estos defienden el cumplimiento de las acciones y operaciones de una forma sistemática.

La evaluación realza el carácter educativo y de autorregulación del proceso ya que en la medida que el estudiante se retroalimenta, permite modularlo. (Anexo No. 05)

2.5 Orientaciones metodológicas para la formación de la habilidad programar en las asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales.

El nivel metodológico del colectivo de profesores que garantiza la instrucción y la educación en la formación de los Ingenieros en Sistemas Computacionales, debe tener didácticamente una acción colegiada y sistemática para influir en el modo en que se enseña y aprenden a programar los estudiantes en las asignaturas de programación de computadoras en los primeros años de la carrera, acorde con los objetivos de cada una de estas y la interrelación de sus invariantes en contenidos, por lo que sus profesores deben prepararse para este propósito.

Se presenta a continuación, un conjunto de indicaciones que facilitan la aplicación del Sistema de Tareas Docentes Integradoras para los estudiantes:

- Analizar el Modelo o Perfil Profesional de la carrera de Ingeniería en Sistemas Computacionales y precisar la implicación curricular que tienen las asignaturas de programación de computadoras de los primeros años, en el modo de actuación del ingeniero en sistemas computacionales (Anexo No. 01) y (Anexo No. 03).

- Valorar el sistema de habilidades de las asignaturas Programación I, II, III y Orientada a Objetos, para identificar aquellas que tributan al modo de actuación del ingeniero en sistemas computacionales.

- Seleccionar los métodos, formas, medios de enseñanza más adecuados de acuerdo con las características del contenido a estudiar en cada asignatura de programación.

- Orientar con precisión los métodos, técnicas y procedimientos propios de las asignaturas de programación de computadoras en los primeros años, para resolver ejercicios, situaciones problémicas reales o simuladas, proyectos integradores mediante la construcción de un programa ejecutado en computadora.

- Sistematizar para cada tipo de tarea, la revisión bibliográfica de los contenidos estudiados y su gradual interrelación.

- Desarrollar en los estudiantes el pensamiento analítico, crítico, la iniciativa creadora para que lleguen a conclusiones por sí mismos.

- Propiciar en el estudiante escenarios para el debate de argumentos, con el fin de defender puntos de vista.

- Aplicar los fundamentos teóricos de la programación de computadoras en sus dos paradigmas: estructurado y orientado a objetos.

- Lograr que el sistema de evaluación sea sistemático e integrador, sobre las bases de sus niveles de sistematicidad: frecuente, parcial y final.

- Utilizar el análisis científico como el expresado en este trabajo que posibilite determinar el nivel de implicación que tienen las habilidades en el modo de actuación del ingeniero en sistemas, para determinar el tipo de habilidad a formar en los estudiantes de la carrera de Ingeniería en Sistemas Computacionales.

- Determinar la concepción holística de las categorías del proceso docente educativo que están presentes en el análisis y tratamiento de la interrelación de las asignaturas.

- Estructurar el Sistema de Tareas Docentes Integradoras soportado en un sistema de acciones y operaciones que posibilite la formación de la habilidad *programar* en las diferentes etapas didácticas según la planificación del profesor.

- Considerar las etapas didácticas propuestas en este sistema y su implementación en cada una de las invariantes funcionales de las asignaturas para adaptarlas y aplicarlas en la formación de la habilidad *programar*.

- Redimensionar los métodos, medios y formas de enseñanza, en función de los objetivos y la interrelación de contenidos que precisen la formación de la habilidad *programar*.

Conclusiones parciales del Capítulo.

La concepción didáctica del Sistema de Tareas Docentes Integradoras para la formación de la habilidad *programar* se concibió partiendo del análisis de la interacción dialéctica de los componentes del proceso docente educativo con un enfoque holístico en la interrelación de contenidos de las asignaturas: Programación I, II, III y Orientada a Objetos de los primeros años en la carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil y sus etapas didácticas: Orientación, Ejecución y Evaluación que se estructuraron en el sistema de acciones y operaciones, para que el estudiante incursione por los niveles de asimilación de Familiarización, Reproducción, Producción y Creación del conocimiento con un enfoque problémico. Se presentan las orientaciones metodológicas para aplicar la propuesta en la práctica pedagógica, desde una perspectiva sistémica e intencional de modo que se contribuya a la formación de la habilidad *programar* y se mejore el desempeño del estudiante en la disciplina de programación de computadoras de los primeros años de la carrera.

Capítulo III: Análisis de los resultados de los métodos utilizados para la formación de la habilidad *programar* en las asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil.

En el presente capítulo se presenta el análisis del resultado de los métodos utilizados en la concepción y la implementación del Sistema de Tareas Docentes Integradoras, se revelan las regularidades del proceso de triangulación de los resultados realizada a partir de la aplicación de métodos como la observación y la encuesta. Se precisan los principales resultados de la validación por el método de criterio de expertos y se exponen los resultados de su aplicación en los estudiantes de las asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil.

3.1 Análisis de la formación de la habilidad *programar* en los estudiantes mediante la el Análisis Documental y la Observación.

La Análisis Documental y la Observación directa, abierta, sistemática y participativa, como métodos de recolección de datos permitió levantar información sobre las características contextualizadas de la formación de la habilidad *programar* en los estudiantes de Ingeniería en Sistemas de la Universidad de Guayaquil y detectar sus principales dificultades en la formación de esta habilidad, en las asignaturas de programación de computadoras ubicadas en los primeros años del Plan Curricular de este profesional.

Del diagnóstico efectuado por el autor, a partir del **Análisis Documental** sobre los boletines de matrícula, portafolios de seguimiento docente y balances semestrales, se destacan las siguientes evidencias:

- En 1996 sobre la base de datos de los bachilleres en Ciencias Digitales e Informática, que cursaron por primera vez la asignatura Programación I en la carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil, se demostró que un 65.8% de estudiantes matriculados revelaban desconocimiento de estas asignaturas y de las habilidades de programación de computadoras, y al finalizar el ciclo semestral un 74.5% enfatizó su desinterés y apatía por esta disciplina o área.

- Desde la puesta en marcha de la carrera de Ingeniería en Sistemas Computacionales en 1997 hasta el 2000 se observó como una regularidad del proceso de enseñanza de las asignaturas de programación de computadoras, que independientemente de las modificaciones realizadas a los planes de estudio de las asignaturas de Programación I, II, III y Orientada a Objetos se mantuvieron las insuficiencias en la formación de la habilidad *programar*, determinadas por la carencia de los métodos adecuados para el logro de este fin, sumado a un 80% del claustro de profesores carente de experiencia metodológica.

- Se constató además, que el 58% de estudiantes de esta cohorte, participaron de una metodología de enseñanza aislada a partir de la transmisión incipiente de conocimientos inconexos mediante clases magistrales, de tal forma que la habilidad *programar* no fue desarrollada sistemáticamente e interrelacionada sobre los contenidos de las asignaturas de programación de computadoras en los primeros años de carrera, lo que ocasionó que aproximadamente el 60% de graduados de esta promoción, no puedan insertarse en empresas locales y nacionales de desarrollo de software.

- En el periodo lectivo comprendido entre el 2001 y el 2007 la motivación por el estudio de la carrera de ingeniería de sistemas se incrementó en un 25% respecto a la matrícula del ciclo anterior, por lo que el número de alumnos que ingresó a esta carrera en la Universidad

de Guayaquil alcanzó una matrícula de 3800 estudiantes, amén de contar con una amplia cobertura en equipos, se implementaron 2 nuevos laboratorios de cómputo.

- No obstante, los portafolios de seguimiento docente en clase, reveló un débil cumplimiento del 40% de los objetivos educacionales trazados en los planes de estudios de las asignaturas de programación de computadoras en los primeros años, implicado en la deficiente formación de habilidades profesionales, específicamente para *programar*, ya que este incremento exponencial de matrícula trajo como consecuencia el incremento proporcional del claustro docente, en donde 8 de cada 10 profesores eran noveles e inexpertos, con carencia de preparación y con deficiencias metodológicas. Además de que estos priorizaron en el proceso educativo, las reglas de sintaxis del lenguaje de programación y menospreciaron la comprensión de conceptos y la formación de habilidades del pensamiento intrínsecas a la lógica de programación.

A partir del 2008 a la fecha, el Consejo de Facultad y el Sistema Nacional de Nivelación y Admisión del Ecuador permitió iniciar estudios de Ingeniería en Sistemas Computacionales en la Universidad de Guayaquil, a bachilleres de cualquier especialización y no exclusivamente de Ciencias Digitales e Informática, como fue un requisito del perfil de ingreso en sus inicios, lo que posibilitó el tratamiento de grupos heterogéneos de estudiantes e impidió que con la misma estructuración metodológica la formación de la habilidad *programar* tenga el mismo nivel de asimilación, comprensión y aplicación, pues los bachilleres genéricos no tenían conocimientos precedentes de la materia, en relación con dicha habilidad.

- En esta etapa se evidencia además, que la asignatura Programación I fue priorizada dentro del pensum curricular como *Fundamentos de la Lógica de Programación*, por su importancia en la formación y desarrollo de la habilidad *programar*.

- En los dieciséis años de vigencia de la carrera de Ingeniería en Sistemas Computacionales (CISC) de la Universidad de Guayaquil los resultados del aprovechamiento académico en las asignaturas de programación de computadoras en los primeros años no son halagadores y revelan su comportamiento creciente del 40% de reprobación y del 6% de deserción estudiantil como se muestra:

Tabla No. 01: Estadística Académica (CISC) 1997 – 2014 (Elaboración Propia)

	AREA DE PROGRAMACIÓN	
PERIODO	**% Reprobados**	**% Desertados**
1997- 2000	35%	5%
2001- 2007	36%	3%
2008- 2014	39%	6%

El bajo nivel de aprovechamiento académico de los estudiantes de las asignaturas del área de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales, se demuestra en un promedio general de 4.6 sobre 10 con proyección a descender progresivamente en el tiempo, si no se presentan intervenciones de solución significativas desde la didáctica para mejorar la formación de la habilidad *programar.*

Del diagnóstico realizado por la **Observación directa, abierta, sistemática y participativa,** se evidencian las siguientes regularidades:

- Ante las dificultades en la enseñanza - aprendizaje de la programación de computadoras se han propuesto numerosas soluciones sin que ninguna haya resultado realmente efectiva. A los problemas de aprendizaje de los estudiantes se suma la falta de un estudio profundo de las habilidades que deben adquirir, reduciendo muchas veces al dominio de las reglas de

sintaxis de un lenguaje de programación e ignorando el marco conceptual de la lógica de la programación.

- Con respecto a la *Concepción del problema*, el autor ha notado que es muy importante tener bien claro el problema que se quiere resolver antes de proceder a resolverlo; los alumnos muchas veces fallan porque leen el problema rápido, de manera descuidada, sin poner atención a los detalles, sin preocuparse por comprender completamente el problema; lo que los lleva muchas veces a resolver un problema diferente del que se les plantea.

- En la etapa de *Diseñar el algoritmo solución*, muchos alumnos generalmente ponen en práctica la primera idea que se les ocurre, sin estar seguros de cómo van a continuar la solución del problema o si esa era la mejor manera de enfrentar el problema.

Finalmente con respecto a *Depurar el programa* o comprobar la solución obtenida, de la observación el autor conjetura que los alumnos generalmente prueban el programa con los datos que aparecen como ejemplo en el planteamiento del problema y si funciona adecuadamente suponen que el programa ya es correcto.

- El proceso de enseñanza de la programación: Fenomenológico, Reproductivo por imitación, Tradicionalista y Esquemático basado en las estructuras sintácticas de un lenguaje y no en los fundamentos de la lógica de programación.

- Incipiente motivación en los estudiantes hacia el aprendizaje de la programación.

- No se propicia el control y autocontrol de la asimilación de contenidos.

- Insuficiente comprensión de la manera en que la programación de computadoras se relaciona con su actividad profesional del Ingeniero de Sistemas.

- Ausente interrelación de contenidos en las asignaturas del área de programación de computadoras en los primeros años del currículo del Ingeniero de Sistemas.

- Débil aplicación de contenidos de la programación en la práctica.

- No se brindan escenarios que propicien que el estudiante busque independientemente nueva información sobre programación de computadoras, necesaria para vencer la contradicción fundamental del proceso educativo de este profesional.

- No se contribuye a la integración de los componentes académico, laboral e investigativo.

- Necesidad de una metodología que forme, consolide y desarrolle las habilidades profesionales y en particular la habilidad *programar* en las asignaturas de programación de computadoras en los primeros años del currículo.

- Claustro de profesores noveles e inexpertos, carentes de formación docente y poseedores de un sistema incompleto de conocimientos teóricos y metodológicos de las asignaturas de programación de computadoras.

Estas regularidades del diagnóstico posibilitó concentrar la atención de este autor en las insuficiencias que tiene la formación de la habilidad *programar* necesaria para configurar el modo de actuación profesional del Ingeniero de Sistemas y proponer alternativas de solución didácticas al problema.

3.2 Análisis de los resultados de la validación de la metodología por el método de criterio de expertos.

Esta metodología permitió extraer la información de un grupo heterogéneo de expertos mediante una lógica sistemática e iterativa de juicios de opiniones hasta llegar a un acuerdo, en este proceso se trató de evitar las influencias de individuos o grupos dominantes y al mismo tiempo existió la retroalimentación facilitando el acuerdo final.

3.2.1. Selección y caracterización de los de expertos.

Uno de los problemas principales es decidir quiénes son los expertos o conocedores del tema a analizar. Los expertos pueden ser especialistas internos o externos. Se entiende por experto al individuo, con un elevado nivel de calificación en una esfera del saber, capaz de

ofrecer valoraciones conclusivas de un problema en cuestión con un máximo de competencia.

Se confeccionó un listado inicial de 15 profesionales, posibles de cumplir los requisitos para ser expertos en la materia de programación de computadoras, y su disposición previamente consultada para participar.

Se realizó una valoración sobre el nivel de experiencia que poseen, evaluando de esta forma los niveles de conocimientos que tienen sobre programación de computadoras. Se les aplicó una encuesta para una autoevaluación de los Niveles de Información y Argumentación que tienen sobre el tema en cuestión. (Anexo No. 06)

En la primera pregunta se les pide que marquen con una X , en una escala creciente del 1 al 10, el valor que se corresponde con el grado de conocimiento o información que tienen sobre la habilidad *programar*.

A partir de aquí se calcula el *Coeficiente de Conocimiento o Información (Kc)*, a través de la siguiente fórmula: $K_c = n * 0{,}1$; donde:

K_c : Coeficiente de Conocimiento o Información

n : Rango seleccionado por el experto

La segunda pregunta de la encuesta permitió valorar un grupo de aspectos que influyen sobre el nivel de argumentación o fundamentación del tema a estudiar. Se determinaron los aspectos de mayor influencia. Se tomaron los valores reflejados por cada experto en la tabla para contrastar con los valores de la Tabla Patrón. (Anexo No. 07).

Los aspectos que influyen sobre el nivel de argumentación o fundamentación del tema a estudiar permitieron calcular el *Coeficiente de Argumentación* (K_a) de cada experto:

$$K_a = \sum_{i=1}^{6} n_i \; = \; (n_1 \; + \; n_2 \; + \; n_3 \; + \; n_4 \; + \; n_5 \; + \; n_6)$$

Donde:

K_a : Coeficiente de Argumentación

n_i : Valor correspondiente a la fuente de argumentación i (1 hasta 6)

Una vez obtenido los valores del *Coeficiente de Conocimiento* (K_c) y el *Coeficiente de Argumentación* (K_a) se calculó el valor del *Coeficiente de Competencia* (K) que finalmente es el coeficiente que determina en realidad qué experto se toma en consideración para trabajar en esta investigación. Este coeficiente (K) se calcula de la siguiente forma:

$$K = \frac{1}{2}\left(K_c + K_a\right)$$

Dónde: K : Coeficiente de Competencia K_c : Coeficiente de Conocimiento

K_a : Coeficiente de Argumentación

Posteriormente obtenidos los resultados se analizan de la manera siguiente:

$0{,}8 \leq K \leq 1{,}0$ Coeficiente de Competencia Alto

$0{,}5 \leq K < 0{,}8$ Coeficiente de Competencia Medio

$K < 0{,}5$ Coeficiente de Competencia Bajo

Se utilizan para la consulta, expertos con Coeficiente de Competencia *Alta*. Para la elaboración del listado de expertos se tuvo en cuenta el *Coeficiente de Competencia*, pero también se analizó la calidad de cada uno de ellos: su conducta activa, su juicio autocrítico, su ética en la discusión, su creatividad y disposición en la solución del problema, su

capacidad de análisis para resolver de forma eficiente, que exija un criterio y enfrente marcos tradicionales o actuales, además su posibilidad real de participación. Su condición profesional, investigaciones y publicaciones realizadas, categoría científica y docente, responsabilidad que desempeña y conocimiento relevante del tema de investigación.

Se seleccionan como expertos 13 profesionales con competencia *Alta*. En el grupo de expertos se incluyen profesores universitarios con amplia experiencia.

Los expertos seleccionados son profesionales en el área de desarrollo de software vinculados a la empresa ecuatoriana, el 85 % son además de ingenieros de sistemas también máster en sistemas de información gerencial y el 15 % tienen solamente Tercer Nivel de Educación Superior en Ingeniería en Sistemas Computacionales. Con relación a la categoría docente, 8 son titulares y 5 contratados, todos poseen más de 10 años de experiencia en la docencia como profesores universitarios de programación y han dirigido tesis de pregrado relacionadas con el tema de esta investigación. (Anexo No. 08)

3.2.2 Método Delphi para validar la concepción del Sistema de Tareas Docentes Integradoras para los estudiantes de programación de computadoras en los primeros años de Ingeniería en Sistemas Computacionales.

Después de seleccionado el grupo de expertos se procesan sus criterios y opiniones para validar la propuesta sustentada en sus conocimientos, investigaciones, experiencia, estudios bibliográficos. Se aplica para este análisis el *Método Delphi* de diagnóstico cualitativo, desarrollado por un grupo de investigadores norteamericanos, teniendo en cuenta la utilización sistemática de valoraciones intuitivas de un grupo de expertos para obtener consenso de opiniones informadas. Este método se destaca fundamentalmente por el anonimato, la retroalimentación y la respuesta estadística de grupo.

Según (Cortés & Iglesias, 2005) este método exige la utilización sistemática del juicio intuitivo de un grupo de expertos para obtener consenso de opinión.

La particularidad fundamental de este método consiste en sostener un diálogo anónimo entre el grupo de profesionales considerados expertos en la temática que se está abordando, los que son consultados de manera individual mediante un cuestionario que se les aplica para obtener un consenso a partir de las valoraciones subjetivas de ellos.

El *Método Delphi* aplicado en esta investigación permitió que los expertos evalúen criterios, previamente seleccionados por el investigador (facilitador) en determinadas categorías de evaluación a *Escala de Likert*. De esta forma los criterios quedan evaluados por los expertos en forma significativa.

Se contó con los 13 expertos seleccionados y 10 criterios para validar el Sistema de Tareas Docentes Integradoras en los estudiantes de programación de computadoras de los primeros años de carrera. Se consideraron 5 rangos de valoración: Inadecuado (I), Poco Adecuado (PA), Adecuado (A), Bastante Adecuado (BA) y Muy Adecuado (MA). (Anexo No. 09)

El procedimiento que se ejecutó, fue:

Se entregó a cada experto un instrumento tipo plantilla con los aspectos a evaluar y los rangos de evaluación que pueden seleccionar para validar el Sistema de Tareas Docentes Integradoras.

Los aspectos a evaluar por los expertos fueron los siguientes:

- Pertinencia de las etapas didácticas declaradas en el Sistema de Tareas.
- Participación y protagonismo de los estudiantes evaluados en el proceso.
- Aplicabilidad del Sistema de Tareas Docentes en el contexto de las carreras de ingeniería de sistemas en las universidades ecuatorianas.

- Pertinencia de los campos y parámetros que se proponen, contribución al carácter sistémico y procesal de la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación de computadoras en los primeros años de su carrera.

- Posibilidad de que la información que se obtiene sobre la formación de la habilidad programar sea fiable y válida, transformación del desempeño del estudiante y de los modos de actuación de los estudiantes evaluados en el proceso.

- Posibilidad de generalización del Sistema de Tareas Docentes Integradoras. Asequibilidad y Flexibilidad del Sistema de Tareas Docentes Integradoras.

A partir de las categorías de evaluación que son: Inadecuado (I), Poco Adecuado (PA), Adecuado (A), Bastante Adecuado (BA) y Muy Adecuado (MA), los expertos asignaron un valor numérico del 1 al 5 en el mismo orden de ponderación de menor a mayor.

De esta forma, el facilitador construyó la *Tabla de Frecuencia Observada* (Anexo No. 10), donde solo se registra la cantidad de expertos que marcó ese rango de valoración.

Luego, se obtuvo la *Tabla de Frecuencia Acumulada* por cada aspecto (Anexo No. 11), a partir de la suma acumulada de las columnas de frecuencia inmediata anterior.

Posteriormente se elaboró la *Tabla de Frecuencia Relativa Acumulada* obtenida al dividirse el valor de cada frecuencia acumulada por la cantidad de expertos, en este caso 13 expertos. El último rango de valoración se elimina pues se están buscando cuatro *Puntos de Corte* PC, pues son cinco rangos de valoración. (Anexo No. 12).

A partir de lo anterior, se obtuvo el valor de la imagen *z*, que corresponde a cada *Frecuencia Relativa Acumulada* mediante la Tabla de Distribución Normal Inversa Estandarizada. Las indagaciones empíricas más frecuentes según la estadística no paramétrica responden a diferentes pruebas estadísticas, el criterio de expertos se procesa mediante la función binomial. (Anexo No. 13)

3.2.3 Criterios de expertos. Resultados del Método Delphi.

Realizados los cálculos correspondientes al Método Delphi para la evaluación de los criterios, se conjeturaron las siguientes conclusiones:

Los *Puntos de Corte PC* se obtienen a partir de los valores probabilísticos de la *Distribución Normal Inversa Estandarizada* divididos respectivamente por cada aspecto entre la sumatoria por rango de valoración entre número de aspectos a evaluar.

$$\text{Puntos de corte} = \frac{Sumatoria\ por\ Rango\ de\ Valoración}{No.\ de\ Aspectos\ a\ Evaluar}$$

Tabla No. 02: Resultados de Puntos de Corte de la Validación Expertos (Elaboración propia)

PC1: Inadecuado = -3.09	**PC2**: Poco Adecuado = -3.09	**PC3:** Adecuado = -1.43
PC4: Bastante Adecuado = -0.38	**PC5:** Muy Adecuado = valores superiores a C4	

La sumatoria de todos los puntos de corte por aspectos a evaluar es – 80.04

Los puntos por cada aspecto se obtienen a través de la diferencia de N – P, donde:

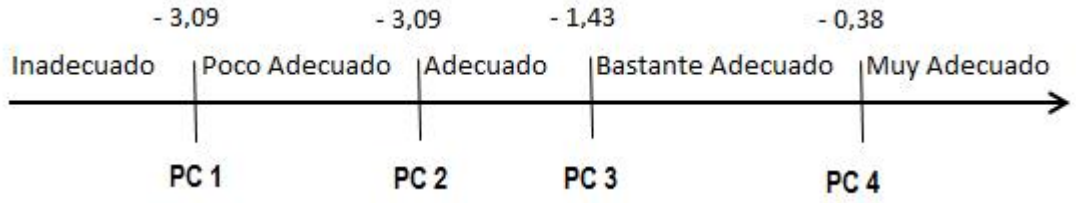
$$N = \frac{Sumatoria\ de\ la\ Suma\ por\ Aspectos}{No.\ Rango\ de\ Valoración\ x\ No.\ Aspectos} \qquad P = \text{Pr}omedio\ por\ Aspectos$$

Tabla No. 03: Valoración de Criterios con Valoración Promedio N-P (Elaboración Propia)

Criterios Evaluados por los Expertos	**Valor Promedio de los Expertos a cada *N-P* alternativo**
[1]Sistematicidad de las cuatro invariantes funcionales para la formación de la habilidad programar: *Concebir el problema, Diseñar el Algoritmo Solución; Traducir lo a un lenguaje de programación y Depurarlo.*	0.46
[2]Pertinencia de las etapas didácticas declaradas en el Sistema de Tareas Docentes Integradoras	0.30
[3]Aplicabilidad del Sistema de Tareas Docentes Integradoras en el contexto de las carreras de Ingeniería en Sistemas Computacionales de las universidades ecuatorianas	0.40
[4]El Sistema de Tareas Docentes Integradoras propicia la participación y el	0.13

protagonismo de los estudiantes en el proceso.	
[5]Pertinencia de los acciones y operaciones que se proponen en las tareas.	1.76
[6]Contribución a la formación de la habilidad *programar* desde la integración sistémica e interrelacionada de los contenidos esenciales de las asignaturas de programación de computadoras en los primeros años de la carrera.	0.36
[7]El Sistema de Tareas Docentes Integradoras propicia en los estudiantes la transformación del desempeño en la habilidad *programar* a partir de los modos de actuación del Ingeniero de Sistemas.	0.38
[8]Flexibilidad del Sistema de Tareas Docentes Integradoras para la dosificación y tratamiento individual de los niveles de asimilación de contenidos de los estudiantes de programación de computadoras.	0.69
[9]Asequibilidad del Sistema de Tareas para la evaluación de la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación de computadoras en los primeros años de su carrera.	0.22
[10]Posibilidad de generalización del Sistema de Tareas Docentes Integradoras para la formación de habilidades profesionales en los estudiantes de Ingeniería en Sistemas Computacionales.	0.14

Estos puntos se ubican en un *Rayo Numérico* junto con los *Puntos de Corte* PC y determinan el rango de valoración para cada criterio.



La ubicación de los *N-P valores* de cada criterio en el gráfico del rayo numérico con los *Puntos de Corte* PC, demuestra el nivel de importancia por consenso, que le otorgan los expertos a cada uno de los 10 criterios.

Como puede observarse en la tabla anterior, todos los *N-P valores* calculados para los 10 criterios son positivos y por lo tanto superiores al máximo *Punto de Corte* PC4 (-0.38) marcado en la posición más a la derecha en el rayo numérico, por lo que corresponde al rango de *Muy Adecuado*.

De este análisis cualitativo-cuantitativo de los resultados del Método Delphi, acerca de los 10 criterios previamente seleccionados por el autor y valorados en 5 categorías por los 13 expertos, se llega a conclusión que todos los aspectos considerados en la propuesta, alcanzan un nivel *Muy Adecuado*, es decir expresan todas y cada una de las propiedades, consideradas como componentes esenciales para determinar al Sistema de Tareas Docentes Integradoras como una propuesta de calidad para la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales.

3.3 Determinación del tamaño de la muestra para la aplicación del Sistema de Tareas Docentes Integradoras en la carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil.

El Sistema de Tareas Docentes Integradoras concebida para la formación de la habilidad *programar* se aplicó en los cursos de Programación I, II, III y Orientada a Objetos en la Carrera de Ingeniería en Sistemas Computacionales de la Facultad de Ciencias Matemáticas y Físicas de la Universidad de Guayaquil durante el 2do. Ciclo Lectivo 2013 - 2014.

De una población total de 14 profesores y 65 estudiantes distribuidos como se muestra en las siguientes tablas:

Tabla No. 04: Población de estudiantes (Elaboración propia)

No.	Denominación	Tamaño Poblacional
1	Estudiantes de Programación I	28
2	Estudiantes de Programación II	14
3	Estudiantes de Programación III	13
4	Estudiantes de Programación O. O.	10
	Total	65

Tabla No. 5: Población de profesores (Elaboración propia)

No.	Denominación	Tamaño Poblacional
1	Profesores de Programación I	6
2	Profesores de Programación II	4
3	Profesores de Programación III	3
4	Profesores de Programación O. O.	1
	Total	14

Mediante la aplicación de la Teoría de Muestreo Probabilístico, se determinó el tamaño de la muestra aleatoria, a través del modelo para proporciones en(Cortes & Iglesias, 1998):

$$n = \frac{\left(\frac{Z_{\sim/2}}{d}\right)^2 p(1-p)}{1+\frac{1}{N}\left(\frac{Z_{\sim/2}}{d}\right)^2 p(1-p)-\frac{1}{N}}$$, de donde:

n : Tamaño de la muestra
$Z_{\sim/2}$: Nivel de Confianza
d : Error de Estimación de la media muestral respecto a la poblacional
p : Proporción de elementos que cumplen la condición
N : Tamaño poblacional

El autor estimó los parámetros del modelo al 95% de Nivel de Confianza equivalente en la Tabla de Distribución Normal Estandarizada de 1.96, el 0.05 de Error de Estimación en la media muestral y un valor de nivel de Proporción del 50% de elementos muestrales que cumplen la condición de máxima proporción en los cursos de programación de computadoras en los dos primeros años de la carrera.

Se calculó el modelo para el tamaño de muestra de cada curso de y se seleccionaron al azar los elementos muestrales en las siguientes proporciones:

Tabla No. 06: Muestra de estudiantes (Elaboración propia)

No.	Denominación	Tamaño de la Muestra
1	Estudiantes de Programación I	24
2	Estudiantes de Programación II	12
3	Estudiantes de Programación III	10
4	Estudiantes de Programación O. O.	6
	Total	52

Tabla No. 07: Muestra de profesores (Elaboración Propia)

No.	Denominación	Tamaño de la Muestra
1	Profesores de Programación I	5
2	Profesores de Programación II	3
3	Profesores de Programación III	2
4	Profesores de Programación O. O.	1
	Total	11

Para la aplicación del Sistema de Tareas Docentes Integradoras en clase. Se tuvo en cuenta la representatividad en la muestra de estudiantes del 2do. Ciclo Lectivo 2013-2014 y de la planta de profesores de esta área en los primeros años de la carrera.

3.4 Análisis de los resultados de la aplicación del Sistema de Tareas Docentes Integradoras en la carrera de Ingeniería en Sistemas de la Universidad de Guayaquil y su validación mediante la triangulación de los resultados de los métodos.

A la muestra de estudiantes y profesores determinada en el epígrafe anterior, se le aplicaron encuestas que permitieron recabar información, acerca del Sistema de Tareas Docentes Integradoras aplicado en la Universidad de Guayaquil. Estos instrumentos dimensionaron los aspectos más relevantes para el análisis y reflexión sobre el tema de estudio:

Encuesta dirigida a los estudiantes (Anexo No. 14):

- ¿Considera que los contenidos de las asignaturas de programación que usted necesita para aprender a programar están presentes en el Sistema de Tareas que se propone?

- ¿El Sistema de Tareas Docentes está elaborado con tareas de carácter integrador que interrelacionan los contenidos más esenciales de las asignaturas de programación de computadoras de los primeros años de la carrera de Ingeniería en Sistemas Computacionales?

- ¿La sistematización de tareas en armonía al resto de las actividades del aula permitió una formación más integral de la habilidad *programar?*

- ¿Las tareas integradoras del sistema están estructuradas a partir del nivel de asimilación de los estudiantes desde los contenidos más sencillos a lo más complejos?

- ¿La aplicabilidad del Sistema de Tareas Docentes integradoras propicia escenarios de práctica y consolidación de conocimientos de los estudiantes?

- ¿La asequibilidad de la etapa didáctica de Orientación y Familiarización de cada tarea integradora, posibilitó el incremento de motivación e interés por aprender a programar?

- ¿El Sistema de Tareas Docentes Integradoras prioriza más que la sintaxis de un lenguaje de programación, el aprendizaje de los fundamentos de la lógica de la programación?

- ¿La factibilidad del sistema de tareas propuesto en la clase, contribuyó a mejorar sus calificaciones y el aprovechamiento académico en las asignaturas de programación de computadoras?

Para el análisis de resultados de las encuestas a estudiantes se codifican y seleccionan del cuestionario anterior las preguntas en términos de las Etapas Didácticas de Orientación, Ejecución y Evaluación:

Tabla No. 8: Códigos de Preguntas a Estudiantes para Análisis de Etapas Didácticas (Elaboración Propia)

Preguntas de la Etapa de Orientación	**Código**

- ¿Considera que los contenidos de las asignaturas de programación que usted necesita para aprender a programar están presentes en el Sistema de Tareas que se propone?	e.3.1
- ¿La asequibilidad de la etapa didáctica de Orientación y Familiarización de cada tarea integradora, posibilitó el incremento de motivación e interés por aprender a programar?	e. 3.6

Preguntas de la Etapa de Ejecución	**Código**
- ¿El Sistema de Tareas Docentes está elaborado con tareas de carácter integrador que interrelacionan los contenidos más esenciales de las asignaturas de programación de computadoras de los primeros años de la carrera de Ingeniería en Sistemas Computacionales?	e.3.2
- ¿La sistematización de tareas en armonía al resto de las actividades del aula permitió una formación más integral de la habilidad *programar?*	e. 3.3
- ¿La aplicabilidad del Sistema de Tareas Docentes integradoras propicia escenarios de práctica y consolidación de conocimientos de los estudiantes?	e. 3.5
- ¿El Sistema de Tareas Docentes Integradoras prioriza más que la sintaxis de un lenguaje de programación, el aprendizaje de los fundamentos de la lógica de la programación?	e. 3.7

Preguntas de la Etapa de Evaluación	**Código**
- ¿Las tareas integradoras del sistema están estructuradas a partir del nivel de asimilación de los estudiantes desde los contenidos más sencillos a lo más complejos?	e. 3.4
- ¿La factibilidad del sistema de tareas propuesto en la clase, contribuyó a mejorar sus calificaciones y el aprovechamiento académico en las asignaturas de programación de computadoras?	e. 3.8

Encuesta dirigida a los profesores (Anexo No. 15)*:*

- ¿El Sistema propuesto propicia el desarrollo de tareas integradoras desde la familiarización, reproducción hasta la producción y creación?

- ¿Con el Sistema de Tareas Docentes Integradoras se percibe en los estudiantes rasgos de una mejor motivación e interés por programar?

- ¿En la propuesta se estructuran tareas integradoras conducentes a potenciar más la lógica de la programación que la enseñanza de reglas sintácticas de un lenguaje de programación?

- ¿Las tareas integradoras demuestran una síntesis de los contenidos esenciales de las asignaturas de programación de computadoras, necesarios para enseñar a programar con cierto nivel de generalización?

- ¿El Sistema de Tareas Docentes Integradoras propician espacios y actividades de práctica y laboratorios de programación?

- ¿Las 4 tareas integradoras que presenta el sistema propuesto posibilitan un vínculo holístico entre lo instructivo y educativo?

Para el análisis de resultados de las encuestas a estudiantes se codifican y seleccionan del cuestionario anterior las preguntas en términos de las Etapas Didácticas de Orientación, Ejecución y Evaluación:

Tabla No. 09: Códigos de Preguntas a Profesores para las Etapas Didácticas (Elaboración Propia)

Pregunta de la Etapa de Orientación	**Código**
- ¿Con el Sistema de Tareas Docentes Integradoras se percibe en los estudiantes rasgos de una mejor motivación e interés por programar?	p.p.2

Pregunta de la Etapa de Ejecución	**Código**
- ¿El Sistema propuesto propicia el desarrollo de tareas integradoras desde la familiarización, reproducción hasta la producción y creación?	p.p.1
- ¿En la propuesta se estructuran tareas integradoras conducentes a potenciar más la lógica de la programación que la enseñanza de reglas sintácticas de un lenguaje de programación?	p.p.3
- ¿El Sistema de Tareas Docentes Integradoras propician espacios y actividades de práctica y laboratorios de programación?	p.p.5

Pregunta de la Etapa de Evaluación	Código
- ¿Las tareas integradoras demuestran una síntesis de los contenidos esenciales de las asignaturas de programación de computadoras, necesarios para enseñar a programar con cierto nivel de generalización?	p.p.4
- ¿Las 4 tareas integradoras que presenta el sistema propuesto posibilitan un vínculo holístico entre lo instructivo y educativo?	p.p.6

Observación a Clases: (Anexo No. 16)

Este método se aplicó a través de su respectiva Guía y se valoraron los siguientes aspectos del Sistema de Tareas Docentes Integradoras, propuesto: Pertinencia, Factibilidad, Aplicabilidad, Asequibilidad, Sistematicidad y el Valor Instructivo- Educativo del Sistema de Tareas Docentes Integradoras propuesto.

La triangulación de los resultados de los métodos, propició el intercambio colaborativo de opinión entre estudiantes y profesores de la Universidad de Guayaquil, y la comparación de estos criterios con las valoraciones recogidas en la observación a clases. La confrontación de estos tres resultados concedió un valor de primer orden en el análisis reflexivo colectivo sobre los aspectos relevantes en la aplicación de la propuesta.

3.4.1 Resultados de las encuestas en la asignatura Programación I.

Luego de procesar los instrumentos de encuestas aplicados en clases teóricas y prácticas a los estudiantes y profesores de Programación I, en las tres Etapas Didácticas de Orientación, Ejecución y Evaluación se obtuvieron los siguientes resultados:

Tabla No. 10: Resultados de Encuestas a Estudiantes de Programación I. (Elaboración propia)

Etapa de Orientación

Cód.	MA	BA	A	PA	D
e.3.1	75%	25%	0%	0%	0%
e. 3.6	71%	29%	0%	0%	0%

Etapa de Ejecución

Cód.	MA	BA	A	PA	D
e.3.2	67%	33%	0%	0%	0%
e. 3.3	83%	13%	4%	0%	0%
e. 3.5	71%	29%	0%	0%	0%
e. 3.7	96%	4%	0%	0%	0%

Etapa de Evaluación

Cód.	MA	BA	A	PA	D
e. 3.4	92%	4%	4%	0%	0%
e. 3.8	100%	0%	0%	0%	0%

Análisis:

En relación con las preguntas e.3.1 y e.3.6 que corresponden a los aspectos de la Etapa de Orientación, el 75% de los estudiantes consideran con un nivel *Muy Adecuado,* que los contenidos de las asignaturas de programación están presentes en la propuesta, y los 24 estudiantes que corresponden al 100% de encuestados afirman con un *Nivel Muy Adecuado y Bastante Adecuado,* que la asequibilidad de las tareas de esta etapa, aumentó su interés por aprender a programar.

Las preguntas e.3.2, e.3.3, e.3.5 y e.3.7 relacionadas a la Etapa de Ejecución, el 85% de los estudiantes de esta asignatura, destacan un nivel *Muy Adecuado,* que el Sistema de Tareas Docentes propuesto está elaborado con tareas de carácter integrador, se propicia la sistematización de los contenidos en las actividades del aula.

La etapa de Evaluación vinculada a las preguntas e.3.4 y e.3.8 reflejaron un 92% y un 100% de nivel *Muy Adecuado*, lo que induce al autor a conjeturar que la propuesta permite que el estudiante transite en tareas de acuerdo con sus niveles de asimilación y contribuya a mejorar el aprovechamiento académico

Tabla No. 11: Resultados de Encuestas a Profesores de Programación I. (Elaboración Propia)

Etapa de Orientación

Cód.	MA	BA	A	PA	D
p.p.2	100%	0%	0%	0%	0%

Etapa de Ejecución

Cód.	MA	BA	A	PA	D
p.p.1	80%	20%	0%	0%	0%
p.p.3	60%	40%	0%	0%	0%
p.p.5	100%	0%	0%	0%	0%

Etapa de Evaluación

Cód.	MA	BA	A	PA	D
p.p.4	80%	0%	20%	0%	0%
p.p.6	60%	40%	0%	0%	0%

Análisis:

La única pregunta p.p2 respecto a la Etapa de Orientación, el 100% de profesores encuestados coincide con un nivel *Muy Adecuado* que el Sistema de Tareas Docentes Integradoras, demuestra rasgos en los estudiantes de un interés positivo a aprender a programar. En las preguntas p.p.1, p.p.3 y p.p.5 resaltan sus resultados, de que el 100% de profesores responden a un nivel *Muy Adecuado y Bastante Adecuado,* respecto al desarrollo de la lógica de programación y el fomento la actividad productiva y práctica mediante las tareas del sistema. El 80% de profesores en la Etapa de Evaluación destaca con un nivel *Muy Adecuado* que el sistema demuestra en el tránsito progresivo en los estudiantes en la integración y síntesis de contenidos esenciales de la programación de computadora.

3.4.2 Resultados de las encuestas en la asignatura Programación II.

Luego de procesar los instrumentos de encuestas aplicados en clases teóricas y prácticas a los estudiantes y profesores de Programación II, en las tres Etapas Didácticas de Orientación, Ejecución y Evaluación se obtuvo los siguientes resultados:

Tabla No. 12: Resultados a Encuestas a Estudiantes de Programación II. (Elaboración Propia)

Etapa de Orientación					
Cód.	MA	BA	A	PA	D
e.3.1	100%	0%	0%	0%	0%
e. 3.6	92%	8%	0%	0%	0%

Etapa de Ejecución					
Cód.	MA	BA	A	PA	D
e.3.2	83%	8%	8%	0%	0%
e. 3.3	92%	0%	8%	0%	0%
e. 3.5	83%	17%	0%	0%	0%
e. 3.7	83%	8%	8%	0%	0%

Etapa de Evaluación					
Cód.	MA	BA	A	PA	D
e. 3.4	100%	0%	0%	0%	0%
e. 3.8	100%	0%	0%	0%	0%

Análisis:

Las respuestas recolectadas en los estudiantes sobre las preguntas e.3.1 y e.3.6 revelan que el 100% con un nivel *Muy Adecuado* sostienen que el Sistema de Tareas Docentes Integradoras, presenta los contenidos esenciales de la asignatura Programación II, mientras que el 92% destaca la asequibilidad de la propuesta frente a un incremento positivo de su interés a la materia. Las preguntas e. 3.4 y e.3.8 correspondientes a la Etapa de Evaluación fueron contestadas con un nivel *Muy Adecuado*, demostrándose que el sistema está bien estructurado y provoca un mejor nivel calificaciones de los estudiantes.

Tabla No. 13: Resultados a Encuestas a Profesores de Programación II. (Elaboración Propia)

Etapa de Orientación					
Cód.	MA	BA	A	PA	D
p.p.2	100%	0%	0%	0%	0%

Etapa de Ejecución					
Cód.	MA	BA	A	PA	D
p.p.1	67%	33%	0%	0%	0%
p.p.3	67%	33%	0%	0%	0%
p.p.5	33%	33%	33%	0%	0%

Etapa de Evaluación					
Cód.	MA	BA	A	PA	D
p.p.4	100%	0%	0%	0%	0%
p.p.6	100%	0%	0%	0%	0%

Análisis:

Se destacan los resultados *Muy Adecuado* de las preguntas p.p.2 de la Etapa de Orientación y las preguntas p.p.4, p.p.6 de la Etapa de Evaluación con un 100%. Los profesores garantizan con un nivel *Muy Adecuado y Bastante Adecuado* la productividad y práctica.

3.4.3 Resultados de las encuestas en la asignatura Programación III.

Luego de procesar los instrumentos de encuestas aplicados en clases teóricas y prácticas a los estudiantes y profesores de Programación III, en las tres Etapas Didácticas de Orientación, Ejecución y Evaluación se obtuvieron los siguientes resultados:

Tabla No. 14: Resultados a Encuestas a Estudiantes de Programación III. (Elaboración Propia)

Etapa de Orientación

Cód.	MA	BA	A	PA	D
e.3.1	90%	10%	0%	0%	0%
e. 3.6	80%	10%	10%	0%	0%

Etapa de Ejecución

Cód.	MA	BA	A	PA	D
e.3.2	100%	0%	0%	0%	0%
e. 3.3	70%	30%	0%	0%	0%
e. 3.5	100%	0%	0%	0%	0%
e. 3.7	90%	0%	10%	0%	0%

Etapa de Evaluación

Cód.	MA	BA	A	PA	D
e. 3.4	100%	0%	0%	0%	0%
e. 3.8	80%	20%	0%	0%	0%

Análisis:

Los estudiantes responden con un 100% de nivel *Muy Adecuado* a las preguntas e.3.2, e.3.4 y e.3.5 garantizando que el Sistema de Tareas Docentes Integradoras propicia la solución de problemas de un nivel de complejidad acorde con su nivel de asimilación. El resto de preguntas concentran sus respuestas en el nivel *Bastante Adecuado,*

Tabla No. 15: Resultados a Encuestas a Profesores de Programación III. (Elaboración Propia)

Etapa de Orientación

Cód.	MA	BA	A	PA	D
p.p.2	100%	0%	0%	0%	0%

Etapa de Ejecución

Cód.	MA	BA	A	PA	D
p.p.1	50%	50%	0%	0%	0%
p.p.3	100%	0%	0%	0%	0%
p.p.5	50%	50%	0%	0%	0%

Etapa de Evaluación

Cód.	MA	BA	A	PA	D
p.p.4	50%	50%	0%	0%	0%
p.p.6	100%	0%	0%	0%	0%

Análisis:

El 100% de los profesores responden con un nivel *Muy Adecuado* a las preguntas p.p.2, p.p.3 y p.p.6 demostrando la contribución de la propuesta en las Etapas de Orientación, Ejecución y Evaluación.

3.4.4 Resultados de las encuestas en la asignatura Programación Orientada a Objetos.

Luego de procesar los instrumentos de encuestas aplicados en clases teóricas y prácticas a los estudiantes y profesores de Programación Orientada a Objetos, en las tres Etapas Didácticas de Orientación, Ejecución y Evaluación se obtuvieron los siguientes resultados:

Tabla No. 16: Resultados a Encuestas a Estudiantes de Programación Orientada a Objetos. (Elaboración Propia)

Etapa de Orientación						Etapa de Ejecución						Etapa de Evaluación					
Cód.	MA	BA	A	PA	D	Cód.	MA	BA	A	PA	D	Cód.	MA	BA	A	PA	D
e.3.1	83%	17%	0%	0%	0%	e.3.2	83%	0%	17%	0%	0%	e. 3.4	100%	0%	0%	0%	0%
e. 3.6	67%	33%	0%	0%	0%	e. 3.3	100%	0%	0%	0%	0%	e. 3.8	100%	0%	0%	0%	0%
						e. 3.5	100%	0%	0%	0%	0%						
						e. 3.7	83%	17%	0%	0%	0%						

Análisis:

En la Etapa de Orientación, el 83% de estudiantes consideran que los contenidos esenciales están presentes en el Sistema de Tareas Docentes Integradoras y un 67% opina que la propuesta por medio de asequibilidad promueve el interés a programar. Las preguntas e.3.3 y e.35 revelan la total aceptación que las tareas del sistema presentan un carácter integrador y de interrelación lo que propicia la aplicabilidad de la propuesta. Los 6 estudiantes opinan con un nivel de *Muy Adecuado* en los criterios e.3.4 y e.3.8 de la Etapa de Evaluación.

Tabla No. 17: Resultados de Encuestas a Profesores de Programación Orientada a Objetos. (Elaboración Propia)

Etapa de Orientación						Etapa de Ejecución						Etapa de Evaluación					
Cód.	MA	BA	A	PA	D	Cód.	MA	BA	A	PA	D	Cód.	MA	BA	A	PA	D
p.p.2	100%	0%	0%	0%	0%	p.p.1	100%	0%	0%	0%	0%	p.p.4	100%	0%	0%	0%	0%
						p.p.3	0%	100%	0%	0%	0%	p.p.6	0%	100%	0%	0%	0%
						p.p.5	100%	0%	0%	0%	0%						

Análisis:

El único profesor encuestado de esta asignatura, concentra sus criterios en los niveles *Muy Adecuado* y *Bastante Adecuado* para las tres Etapas Didácticas de aplicación del Sistema de Tareas Docentes Integradoras.

3.4.5 Análisis de Resultados Generales de la Etapa de Orientación.

Esta fase se caracterizó por ofrecer al estudiante la Base Orientadora para la Acción en las conferencias iniciales, en correspondencia con los objetivos de cada tarea.

Resultados Generales de Encuestas a Estudiantes en la Etapa de Orientación.

Los resultados de las encuestas a estudiantes en el aspecto motivacional reveló que el 84,3% de estudiantes mostró interés positivo hacia el aprendizaje del desarrollo lógico de la programación, es decir, aproximadamente el 90% de la clase estuvo motivada con las acciones y operaciones de familiarización en cada contenido integrador de la tarea y esto constituyó una mejora del desinterés detectado en los estudiantes de los primeros años.

Resultados Generales de Encuestas a Profesores en la Etapa de Orientación.

El 75,6% de profesores encuestados, afirmaron que las acciones de orientación en cada tarea integradora llevaron implícito un proceso de familiarización de los estudiantes de lo que saben y necesitan saber. Desde el nivel reproductivo, notaron que los estudiantes de la Universidad de Guayaquil, al repetir el contenido esencial estructurado en tareas del sistema propuesto, mediante el análisis de problemas de programación resueltos, experimentaron una mejora motivacional al salir de un estadio eminentemente fenomenológico a un escenario de asimilación inicial que les permitió con autonomía: *describir, comparar, caracterizar y analizar* problemas propuestos sencillos.

Resultados Generales de la observación a clases en la Etapa de Orientación.

La observación a clases de esta Fase de Orientación, evidenció un alto grado motivacional y de interés por aprender a programar, pues se cuantificó un nivel de 85,9 % de Asequibilidad *Muy Adecuada* y del 89,1% de Pertinencia *Muy Adecuada*, del sistema propuesto en los estudiantes de programación de computadoras en los primeros años.

La triangulación de los resultados generales en la Etapa de Orientación.

La triangulación de los resultados generó un análisis cualitativo- comparativo de cada fuente sobre los aspectos motivacionales de esta etapa, llegando a un consenso *Muy Adecuado* de criterios respecto a la etapa de familiarización del Sistema de Tareas Docentes Integradoras dictado en conferencias, talleres y laboratorios.

3.4.6 Análisis de Resultados Generales de la Etapa de Ejecución.

Esta etapa se caracterizó por exigir al estudiante en clase la realización de acciones y operaciones propias del nivel de asimilación del conocimiento productivo, presidido por un objetivo de carácter trascendental, en cuyo meollo se justificó la necesidad de la integración del contenido de las asignaturas de programación de computadoras en cada tarea planificada, se propició un alto componente de auto preparación de los estudiantes que los llevó al dominio progresivo de los conocimientos acercándolos gradualmente a formar la habilidad *programar.*

Resultados Generales de Encuestas a Estudiantes en la Etapa de Ejecución.

Los resultados de las encuestas a los estudiantes revelan que en los cursos donde se implementó el Sistema de Tareas Docentes Integradoras, como es el caso de los estudiantes de Programación I, II, III y Orientada a Objetos del 2do. Ciclo 2013-2014 en la Universidad de Guayaquil, de una muestra de la matrícula de 52 estudiantes; 38 alcanzaron un mejor

aprovechamiento académico con calificación de Excelente o Muy Bien luego de resolver situaciones problémicas simuladas y de la vida real, contenidas en la propuesta.

Se determinó en los estudiantes encuestados, que la aplicación del Sistema de Tareas Docentes Integradoras permitió que el 75,4% manifieste que sí aprendió a programar computadoras con mejor eficiencia.

Un 86,3% de la encuestas a estudiantes revelan que las acciones y operaciones planteadas en el sistema de tareas, les permitió recorrer por los conocimientos y habilidades de la programación desde lo sencillo a lo complejo.

Resultados Generales de Encuestas a Profesores en la Etapa de Ejecución.

Los 11 profesores encuestados del área de programación de computadoras de la Universidad de Guayaquil, expresaron criterios muy interesantes con respecto a la formación de la habilidad *programar* mediante el sistema de tareas propuesto, un 91% revela una tendencia de mejores registros de aprovechamiento académico respecto a los dos periodos lectivos anteriores, 8 de los 11 profesores encuestados, destacan la interrelación de contenidos esenciales propuestos en el sistema de tareas integradoras como una vía metodológica para mejorar la formación de la habilidad *programar*, pues en el sistema propuesto se abordan situaciones problémicas acordes con el nivel de asimilación individual en los estudiantes, un 63,9% coincide en que la propuesta garantiza la integración de lo académico, laboral e investigativo en la formación de los estudiantes de ingeniería de sistemas desde los primeros años de estudio.

De los profesores encuestados, el 88.5 % considera que la Factibilidad por etapas del Sistema de Tareas Docentes Integradoras es *Muy Adecuada* y el resto, es decir un 11.5 % manifiestan que es *Bastante Adecuada* y *Adecuada*.

Se calculó que el 93% de profesores que aplicaron este sistema reveló la relación del componente instructivo y educativo en la estructuración de las tareas integradoras y el vínculo que realiza el estudiante desde su asignatura de programación de computadoras, con la práctica y el contexto profesional.

Observaciones Generales a clases en la Etapa de Ejecución.

Esta etapa se caracterizó según la valoración de los aspectos de la observación a clases en el cumplimiento de las acciones y operaciones estructuradas en las Tareas Integradoras de tipo problémico a un Nivel Productivo donde el estudiante tuvo que *valorar, clasificar, interpretar, argumentar, reducir* preferentemente Problemas Propuestos y a un Nivel Creativo en el cual el estudiante debe *determinar, generalizar, aplicar, crear, innovar* soluciones informáticas de Problemas Integradores basados en programación.

Las situaciones problémicas presentadas en el sistema demuestran un nivel del 89,2% de Sistematicidad *Muy Adecuado* y un 95,9% de Aplicabilidad *Bastante Adecuado.*

La triangulación de los resultados generales de los métodos en la Etapa de Ejecución.

La triangulación de los resultados en esta fase, derivados de las encuestas dirigidas a estudiantes, profesores y la observación a clases, posibilitó al autor la valoración *Bastante Adecuada* de la Fase de Ejecución del proceso de formación de la habilidad *programar* mediante la resolución de problemas reales o simulados por parte del estudiante de la Universidad de Guayaquil.

3.4.7 Análisis de Resultados Generales de la Etapa de Evaluación.

En esta fase se valoró el cumplimiento de los objetivos de la propuesta, orientados a controlar el aprendizaje sistemáticamente de forma armónica e integrada del estudiante de la Universidad de Guayaquil, mediante el Sistema de Tareas Docentes Integradoras para la formación de la habilidad *programar.*

Se conciliaron los resultados de encuestas a estudiantes, profesores y se trianguló con la información obtenida por la observación a clases en la Fase de Evaluación y se expresó un juicio de valor definitivo.

Resultados Generales de Encuestas a Estudiantes en la Etapa de Evaluación.

El 91,8% de los estudiantes encuestados consideran que las tareas integradoras del sistema propuesto propician la autoevaluación. No obstante, se evidencia que un porcentaje mínimo del 8.2 % de los estudiantes encuestados no tienen en cuenta en su autoevaluación, los elementos que aparecen en la propuesta, y se supone lo hacen solo con sus propias condiciones como estudiante y escasos elementos sistemáticos relacionados con la formación de la habilidad *programar*, lo que impide que sea un proceso integral.

Resultados Generales de Encuestas a Profesores en la Etapa de Evaluación.

Esta etapa en el Sistema de Tareas Docentes Integradoras según las encuestas a profesores confirman que en un 87,1% las acciones y operaciones de evaluación realzan el carácter educativo y de autorregulación del proceso, ya que en la medida que el estudiante se retroalimenta, permite modularlo.

Observación Generales a clases en la Etapa de Evaluación.

La evaluación sistémica que plantea el Sistema de Tareas Docentes Integradoras permitió constatar por la observación a clases el grado de independencia que va logrando el estudiante de la Universidad de Guayaquil en la ejecución de las tareas integradoras y el progreso en el nivel de dominio de la habilidad *programar*. Lo que demostró el 79,4% de Factibilidad y un 97 % de Valor Instructivo Educativo del sistema propuesto.

La triangulación de los resultados generales de los métodos en la Etapa de Evaluación.

La triangulación de resultados, en esta fase permitió ejecutar la evaluación en el Sistema de Tareas Docentes Integradoras y se demostró su pertinencia *Bastante Adecuada*, según se

logra el nivel de transformación del estudiante de los primeros años en lo que concierne a su habilidad *programar*.

3.5 Conclusiones parciales del Capítulo.

Los resultados obtenidos en el diagnóstico inicial mediante la *Observación directa, abierta, sistemática y participativa,* y *Análisis Documental* del desempeño de los estudiantes de programación de computadoras en sus primeros años de carrera en la Universidad de Guayaquil en el período lectivo 2013 – 2014, posibilitó la elaboración de la propuesta.

Según el *Criterio de los Expertos* consultados, mediante el método Delphi, se determinaron los 10 aspectos que caracterizan la concepción del Sistema de Tareas Docentes Integradoras, la que se demuestra como una propuesta con un nivel *Muy Adecuado* para la formación de la habilidad *programar*, a partir de su factibilidad, pertinencia, aplicabilidad y potencialidades transformadoras.

El análisis de las encuestas, la observación a clases en esta universidad, y la posterior triangulación de los resultados de los métodos, permitió garantizar el rigor científico del Sistema de Tareas Docentes Integradoras y su implementación en la carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil, facilitó la validación del sistema de acciones y operaciones, mediante la aplicación de cada etapa didáctica de Orientación, Ejecución y Evaluación de la formación de la habilidad *programar* en los estudiantes.

Conclusiones Generales

El estudio de los referentes teóricos relacionados con las regularidades de la formación de la habilidad *programar* desde las asignaturas de programación de computadoras, los métodos utilizados en la investigación, la concepción, fundamentación y aplicación del Sistema de Tareas Docentes Integradoras propuesto para la formación de esta habilidad, permiten al autor elaborar las siguientes conclusiones:

- El análisis de la bibliografía y el diagnóstico realizado posibilitó conocer que la formación de la habilidad *programar* no tiene un tratamiento didáctico, por lo que el autor la concibe como la actividad que realiza el estudiante de forma interrelacionada y práctica de la lógica de la programación estructurada y orientada a objetos, implicada en la dimensión de los contenidos esenciales en esta disciplina, para resolver problemas técnicos, desde los primeros cuatro semestres de la carrera de Ingeniería en Sistemas Computacionales.

- Se concibe el Sistema de Tareas Docentes Integradoras como un sistema de acciones y operaciones con enfoque problémico, en donde se sintetizan en tres etapas didácticas: *Orientación, Ejecución y Evaluación,* tareas con un carácter integrador desde la interrelación en los niveles de *Familiarización, Reproducción, Producción y Creación.*

- El Sistema de Tareas Docentes Integradoras elaborado fue sometido a la validación de Criterios de Expertos lo que posibilitó su construcción y se aplicó en la carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil. Los resultados de las encuestas, la observación a clases y la triangulación de estos resultados, permitió validar la propuesta concebida mediante la aplicación de cada tarea, por etapas demostrándose la pertinencia del sistema de acciones y operaciones integradoras concebido y su factibilidad de aplicación, pues contribuye a mejorar la formación de la habilidad *programar*.

Recomendaciones

- Sistematizar las Tareas Docentes Integradoras al resto de las áreas disciplinares de la carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil, y al resto de Universidades y Escuelas Politécnicas que ofertan esta carrera y que se encuentran bajo la rectoría de la Secretaría Nacional de Educación Superior, Ciencia y Tecnología del Ecuador, con el propósito de consolidar desde la Didáctica, la formación de las habilidades del Ingeniero de Sistemas atendiendo a sus necesidades y su contexto social.

- Capacitar a los profesores de la Universidad de Guayaquil, con el contenido de la metodología propuesta, para perfeccionar su formación docente pedagógica en pro de los procesos educativos que dirija.

- Perfeccionar el sistema de acciones y operaciones a partir de la retroalimentación de los resultados de aprendizaje de programación de computadoras en los estudiantes de carreras de corte tecnológico afines a la Ingeniería de Sistemas en la sistematización de la propuesta.

Bibliografía

(2001). *Computing Curricula*. USA: Association Computer Machinery ACM, IEEE Computer Society.

(2003). *Colectivo de autores. Diseño y desarrollo del Currículo*. Cienfuegos, Cienfuegos, Cuba: KRK.

(2005). *Computing Curricula*. USA: Association Computer Machinery ACM, IEEE Computer Society.

(2008). *Computing Curricula*. USA: Association for Computing Machinery, IEEE Computer Society.

(2012). *Misión, Visión y Perfil Profesional del Ingeniero en Sistemas Computacionales de la Universidad de Guayaquil*.

(2013). *Malla Curricular de la Carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil*.

Advine, F., & García, G. (2005). La Tarea Integradora: Eje Integrador Interdisciplinario. (págs. 15-16). La Habana: Ministerio de Educación.

Álvarez de Zayas, C. (2001). *El Diseño Curricular.* La Habana, Cuba: Pueblo y Educación.

Álvarez C. (2002). *La Pedagogía como Ciencia: Epistemología de la Educación.* Sucre, Bolivia: Universidad de San Francisco Javier.

Álvarez, C. (2003). *Metodología del Diseño Curricular.* La Habana, Cuba: Pueblo y Educación.

Beane, J. (1991). *Middle School: The Natural Home of Integrated Curriculum.* USA: Educational Leadership.

Brito Fernández, H. (1984). Hábitos, habilidades y capacidades. *Revista Científico Metodológica "Enrique José Barona"*, 73-78.

Cañedo, C. (2004). *Estrategia Didáctica para Contribuir a la Formación de la Habilidad Profesional Esencial "realizar el paso del sistema real al esquema de análisis" en el Ingeniero Mecánico.* Cienfuegos, Cuba.

Carballosa, A. (2004). *Estrategia Interdisciplinaria para la Enseñanza del Inglés en la Carrera de Estudios Socioculturales.* Cienfuegos, Cuba: UCF.

Caro, S. (2003). *Lógica de Programación y Algoritmos.* Tunja, Colombia: CIPADE.

Cassola, E. (2004). Elaboración de Material Educativo para la Formación de Profesionales en Desarrollo de Software. *Congreso Iberoamericano de Eduación Superior en Computación CIESC - Conferencia Latinoamericana de Informática CLEI.* Perú.

CEAACES. (2011). *Reglamento del Régimen de Acreditación de Carreras*. Quito, Pichincha, Ecuador: Consejo de Evaluación de Autoevaluación Acreditación de la Educación Superior del Ecuador (CEAACES).

CEDEFOP. (14 de Julio de 2010). Recuperado el 8 de Octubre de 2012, de http://www.fi.upm.es/docs/estudios/grado/901_CareerSpace_Profiles.pdf

Collazo, R. (2005). *Tareas de Aprendizaje: Sus exigencias actuales.* México: CEIDE.

Comenius, J. A. (1983). *Didáctica Magna.* La Habana, Cuba: Editorial: Publo y Educación.

Constable, R. L. (2008). *Computer Science: Achievements and Challenges circa 2008.* USA: Addison-Wesley.

Copi, I., & Cohen, C. (2000). *Introducción a la Lógica.* México: Limusa.

Corbí, A. (1998). *Fundamentos de Programación: Metodología.* Alicante, España: Universidad de Alicante .

Corona, L. A. (13 de Octubre de 2006). *El Método Clínico como un Método para el Diagnóstico Médico: Crítica a una concepción vigente.* Recuperado el 12 de Septiembre de 2011, de http://medisur.sld.cu/index.php/medisur/article/view/221pñlo

Corona, L. A. (2010). Sistema de tareas docentes para la formación de la habilidad toma de decisiones médicas mediante el método clínico. *Medisur*, 35-45.

Cortés, M., & Iglesias, M. (1998). *Generalidades sobre la Metodología de Investigación.* Riobamba, Ecuador: UNACH.

Cortés, M., & Iglesias, M. (2005). *Metodología de la Invesigación.* Ciudad del Carmen. México: UNACAR.

Danilov, M. A.; Skatkin, M. N. (1980). *Didáctica de la escuela media.* La Habana: Editorial de libros para la educación.

Davidov, V. V. (1986). *Las Bases Teóricas Metodológicas de las Investigaciones Psicológicas de la Actividad Docente en la formación de escolares.* La Habana, Cuba: Pueblo y Educación.

De Graff, E., & Kolmos, A. (2007). *Managment of Change: Implementation of Problem-Based and Project.Based Learning in Engineering.* USA: Sense Publishers.

Del Río, S. L. (2007). *Técnica de Solución de Problemas utilizando una computadora.* México.

Drake, S. M. (1991). *How our Team Dissolved the Boundaries.* USA: Educational Leadership.

EDUTEKA. (2004). *21 st. Century Skills: Logros indispensables para los estudiantes del Siglo XXI.* Barcelona, España: Eduteka.

Elkonin, D. B. (1986). Las cuestiones psicológicas relativas a la formación de la actividad docente en la edad escolar menor. *Antología de la Psicología Pedagógica y de las edades*, 101-104.

Fariñas, G. (1995). *Estrategias Didácticas.* La Habana, Cuba: Academia.

Fiallo, J. P. (2004). *La Interdisciplinariedad en la Escuela: De la Utopía a la Realidad.* La Habana, Cuba: Ediciones: Instituto de Ciencias Pedagógicas.

Fogarty, R. (1993). *Ten Ways to Integrated Curriculum.* USA: Educational Leadership.

Fuentes, H., & Cols. (1997). *Fundamentos Didácticos para un Proceso de Enseñanza Aprendizaje Participativo.* Santiago de Cuba, Cuba: Universidad de Oriente.

Gallardo, J., & García, C. (8 de Diciembre de 2004). *Apuntes para la Asignatura Informática, Diseño de Algoritmos y Programas.* Recuperado el 9 de Agosto de 2012, de http://www.lcc.uma.es/personal(pepeg/mates

Galperin, P. (1982). *Introducción a la Psicología.* La Habana, Cuba: Pueblo y Educación.

García, L., González, B., & Soto, Y. (2012). Sistema de Tareas Docentes en la Eneñanza del Inglés: En Enfoque Integrador en la Carrera de Medicina. *Universidad y Sociedad*, 23-45.

Gimeno, J. (1995). *Comprender y Transformar las Enseñanzas.* España: Editorial Morata.

Green, L. C. (1991). *Science-Centered Curriculum in Elementary School.* USA: Educational Leadership.

Grishin, D. M. (1965). *Acerca de los tipos y la estructura de las tareas docentes.* Moscú: Soviets-kaia Pedagógica.

Iglesias, M. (1998). La Autopreparación de los estudiantes en los primeros años de la Educación Superior. *Tesis en Opción al Grado Científico de Doctor en Ciencias Pedagógicas*. Cienfuegos, Cienfuegos, Cuba: Universidad de Cienfuegos.

Iñigo, E. (2005). Acercamiento a una propuesta de Relación y Clasificación de Competencias profesionales para la Evaluación del Desarrollo Profesional de los Graduados en Cuba. *Revista Cubana de Educación Superior*, 34-47.

Iranzo, J. (2005). *lógica Simbólica para Informáticos.* México: Alfaomega.

Jenkins, T. (2001). The Motivation of Students Of Programming. *6th. Annual Conference on Innovation and Technology in Computer Science Education.* Canterburry, United Kingdom.

Joyanes, L. (1996). *Fundamentos de Programación Orientada a Objetos, Algoritmos y Estructuras de Datos.* Madrid, España: McGrawHill.

Joyanes, L., Rodríguez, L., & Fernández, M. (2003). *Fundamentos de Programación.* Madrid, España: McGrawHill.

Klingberg, L. (1978). *Introducción a la Didáctica General.* La Habana, Cuba: Pueblo y Educación.

Kölling, M. (1999). The Problem of Teaching Object-Oriented Programming Part.1. *Journal of Object-Oriented Programming,* (págs. 8-15). USA.

Labarrére, G., & Valdivia, G. (1988). *Pedagogía.* La Habana, Cuba: Pueblo y Educación.

Laboda, J., Galimany, J., Peña, R. M., & Gual, A. (1985). *Biblioteca Práctica de la Programación.* Barcelona: Océano.

Lapido, M. (2009). La Formación de las Habilidades Profesionales del Ingeniero Informático: Una necesidad contemporánea. *Escenarios*, 20-35.

Leontiev, A. N. (1982). *Actividad, Conciencia y Personalidad.* La Habana: Pueblo y Educación.

Levine, G. (2001). *Computación y Programación Moderna.* México: Addison-Wesley.

LOES. (2010). *Ley Orgánica de la Educación Superior (LOES)*. Montecristi, Manabí, Ecuador: Registro Oficial.

López, J. (1989). *La orientación como parte de la actividad cognoscitiva de los escolares.* La Habana: Pueblo y Educación.

López, M. (1990). *Saber enseñar a describir, comparar y argumentar.* México: Trillas.

López, M., Whalley, J., Robbins, P., & Lister, R. (2008). *Relationships between reading, tracing and writing skills in introductory programming.* Sidney, Australia: Computing Education Research.

Majmutov, M. (1983). *La enseñanza problémica.* La Habana: Pueblo y Educación.

Maridueña, M. (2014). La habilidad de programación de computadoras: Desde un enfoque concéntrico y radial en el currículo del Ingeniero en Sistemas Computacionales. *Universidad y Sociedad*, 12-14.

Moltó, E. (2009). *Importancia de las tareas educativas y del concepto situación del objeto físico en los cursos dela Física.* Lat. Am. J. Phys. Educ.

Okón, V. (1968). *Fundamentos de la enseñanza problémica.* Moscú: Editorial Instrucción Pública.

Ortiz, A. (2005). *Modelos pedagógicos: Hacia una escuela del desarrollo integral.* Colombia: CEPEDID.

Ortiz, E. (2006). *Retos y Perspectivas del Currículo Integrado.* Puerto Rico: Ediciones: Investigación en la Educación.

Perry, G. (2001). *Absolute Beginner´s Guide to Programming.* Indianapolis, USA: Indiana: Printers.

Petrosvky, A. V. (1985). *Psicología General.* Moscú: Progreso.

Poirier, P. (1997). *La resolución de problemas en la enseñanza.* Canadá: Impresión Ligera.

Ramírez, L. (2010). La tarea integradora, una metodología para su concepción, desarrollo y evaluación. *VII Taller Internacional de la ETP*. Santiago de Cuba.

Reyes, L. M. (2009). Estrategia Didáctica para la Enseñanza de la habilidad programar: Una alternativa para los profesores de los Politécnicos de Informática. *Ciencia, Tecnología y Sociedad*, 10-28.

ROBOCODE. (12 de Enero de 2007). *Robocode: The Open Source Educational Game.* Recuperado el 2 de Mayo de 2008, de http://robocode.sourceforge.net/

Rousseau, J. J. (1970). *Emilo.* México: Porrúa.

Rubinstein, J. L. (1977). *Principios de la Psicología General.* La Habana, Cuba: Pueblo y Educación.

Savin, N. V. (1972). *Pedagogía.* La Habana, Cuba: Pueblo y Educación.

Scaffidi, C., Shaw, M., & Myers, B. (2005). An Approach for Categorizing End User Programmers to Guide Software Engineering Research. *1st. Workshop on End User Software Engineerong WEUSE at the International Conference on Software Engineering*.

SENESCYT. (2013). *Reglamento de Régimen Académico de la Educación Superior del Ecuador*. Quito, Pichincha, Ecuador: Registro Oficial.

SENPLADES. (2013). *Plan Nacional del Buen Vivir.* Quito.

Silvestre, M. (2000). *Enseñanza y Aprendizaje Desarrollador.* México: CEIDE.

Talízina, N. (1987). *La Formación de la Actividad Cognoscitiva de los Escolares* . La Habana, Cuba: EMPES.

Torres, J. (1995). *Globalización e Interdisciplinariedad: El Currículo Integrado.* Madrid: Ediciones Morata.

UNESCO. (2002). *Panorama Educativo de las Américas*. Santiago de Chile, Chile: UNESCO.

Ushinski, K. D. (1957). *Obras completas.* Moscú: Leningrado.

Vars, G., & Beane, J. A. (30 de Noviembre de 2001). *Integrative Curriculum in a Standards-Based World.* Recuperado el 13 de Noviembre de 2004, de http://www.nma.org/research/res-articles-integrated.htm

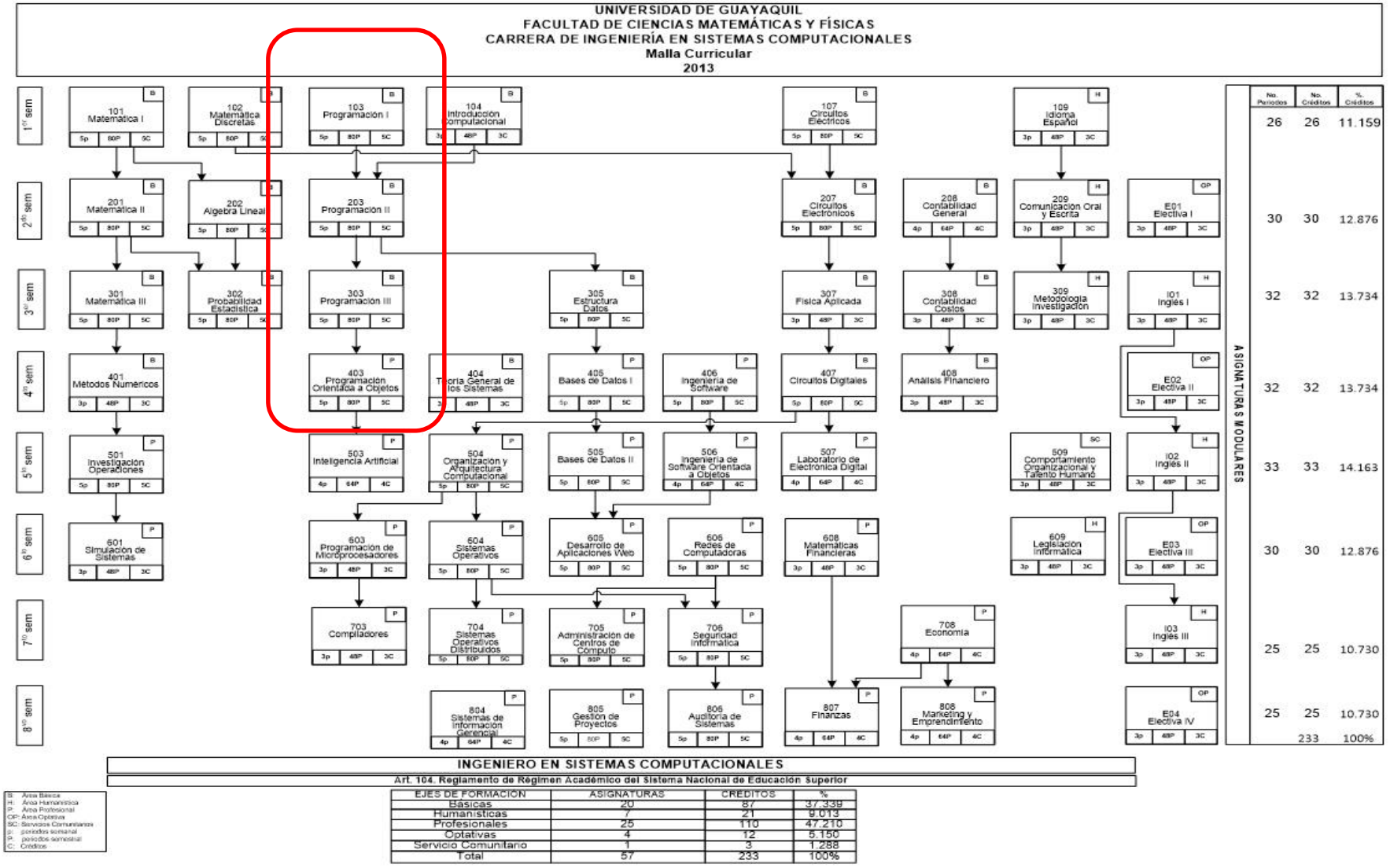
Vigotsky, L. S. (1978). *El Desarrollo de los Procesos Psicológicos Superiores.* Barcelona: Grijalbo.

Zardón, O., & Rojas, R. T. (2013). *La Tarea Docente: Una Vía para la Resolución de Problemas Químicos con Cálculos.* La Habana, Cuba: Pueblo y Educación.

Zhu, H., & Zhou, M. (2009). *Methodology First and Language Second.* USA.

Zilberstein, J. (2005). *Didáctica Desarrolladora desde el Enfoque Histórico Cultural.* México: CEIDE.

Anexo No. 1



INGENIERO EN SISTEMAS COMPUTACIONALES

Art. 104. Reglamento de Régimen Académico del Sistema Nacional de Educación Superior

B: Área Básica
H: Área Humanística
P: Área Profesional
OP: Área Optativa
SC: Servicios Comunitarios
p: períodos semanal
P: períodos semestral
C: Créditos

EJES DE FORMACIÓN	ASIGNATURAS	CREDITOS	%
Básicas	20	87	37.339
Humanísticas	7	21	9.013
Profesionales	25	110	47.210
Optativas	4	12	5.150
Servicio Comunitario	1	3	1.288
Total	57	233	100%

Anexo No. 2

UNIVERSIDAD DE GUAYAQUIL
FACULTAD DE CIENCIAS MATEMATICAS Y FISICAS
CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES

PROGRAMA ANALITICO 2013-2014

1. DATOS GENERALES

ASIGNATURA: **PROGRAMACIÓN I**
CÓDIGO : **103** CRÉDITOS: **5**
PRE-REQUISITO: **PRE UNIVERSITARIO**
PERÍODOS POR SEMANA: **6**
PERÍODOS POR SEMESTRE: **96 PERÍODOS** (16 semanas)

2. DESCRIPCIÓN SINTÉTICA

Al final del curso el alumno será capaz de resolver problemas sencillos de la vida diaria utilizando estructuras de control, funciones definidas por el usuario, arreglos unidimensionales y bidimensionales mediante el lenguaje de programación C.

3. OBJETIVOS

Generales:

- Definir un programa de computadora
- Resolver problemas mediante la construcción de programas.
- Diferenciar los lenguajes de programación por su paradigma
- Conocer y hacer uso de la programación Estructurada.
- Utilizar los pasos para desarrollar un software sencillo.

Específicos:

- Conocer lo tipos de datos predefinidos y hacer uso de cada uno de ellos.
- Definir lo que se entiende por Constantes y Variables y su manera de declararlas y utilizarlas.
- Identificar los diferentes operadores relacionales y boléanos, así como trabajar y hacer programas como expresiones que hagan uso de los operadores.
- Manipular las Estructuras de Control, con decisiones y operadores relaciónales.
- Definir las funciones propias, necesarias para la resolución de problemas y diseñar programas que hagan uso de esas funciones.
- Aplicar las estructuras de datos conocidas como cadenas o arreglos de una dimensión.
- Generalizar los conceptos de arreglos a 2 o más dimensiones.

4. CONTENIDOS PROGRAMATICOS

UNIDAD	NOMBRE	PERIODOS
1	INTRODUCCIÓN A LENGUAJE C (I Parcial)	6
2	TIPOS DE DATOS VARIABLES Y CONSTANTES EN C	4
3	OPERADORES Y EXPRESIONES	4
4	ENTRADAS Y SALIDAS DE DATOS	10
5	ESTRUCTURAS DE CONTROL	24
6	FUNCIONES EN LENGUAJE C (II Parcial)	18
7	ARREGLOS UNIDIMENSIONALES	14
8	ARREGLOS BIDIMENSIONALES	16

UNIDAD 1 **Períodos: 6**

UNIDAD 1	TITULO	Sesión
1.1	Conceptos básicos de programación	1
1.2	Historia de Lenguaje C	1
1.3	Estructura General de un programa	2
1.4	Conceptos de librerías	3

UNIDAD 2 **Períodos: 4**

UNIDAD 2	TITULO	Sesión
2.1	Constantes	4
2.1.1	Concepto de Constantes	4
2.1.2	Declaración de Constantes	4
2.2	Variables	5
2.2.1	Concepto de Variables	5
2.2.2	Declaración de Variables	5
2.2.3	Asignación de valores a Variables	5

UNIDAD 3 **Períodos: 4**

UNIDAD 3	TITULO	Sesión
3.1	Expresiones Aritméticas	6
3.1.1	Operadores Aritméticos	6
3.1.2	Jerarquía de operadores aritméticos	6
3.2	Expresiones Relaciónales	6
3.2.1	Operadores Relaciónales	6
3.2.2	Jerarquía de operadores aritméticos y relacionales	6
3.3	Expresiones Booleanas	7
3.3.1	Operadores de verdad y tablas de verdad	7

UNIDAD 4 **Períodos: 10**

UNIDAD 4	TITULO	Sesión
4.1	Funciones de Entrada (scanf, cin, getch, gets, getchar, getc)	8,9
4.2	Funciones de Salida (printf, cout, putc, putchar, gotoxy, cat)	10,11
4.3	Ejercicios de Aplicación	12

UNIDAD 5 **Períodos: 24**

UNIDAD 5	TITULO	Sesión
5.1	Sentencia If	**13**
5.1.1	IF simple	13
5.1.2	IF Doble	14
5.1.3	IF Múltiple	15
5.1.4	IF Anidado	16
5.2	Sentencia Switch	17
5.3	Sentencia While	18,19
5.4	Sentencia Do/While (Enviar un proyecto que se recogerá al final del primer parcial)	20,21
5.5	Sentencia For	22,23
5.6	Ejercicios de Aplicación	24

UNIDAD 6 **Períodos: 18**

UNIDAD 6	TITULO	**Sesión**
6.1	Introducción a Funciones	25
6.2	Definición de una función	25
6.3	Llamada a una función	25
6.4	Declaración de una función	25
6.5	Funciones para manejo de números aleatorios	26
6.6	Envió de parámetros a una función	27
6.6.1	Parámetros por valor	28
6.7	Funciones para Manejo de Cadenas de Caracteres (strcpy, strcmp, strcmpi, strcat, strlen)	29, 30, 31,32
7	Ejercicios de Aplicación	33

UNIDAD 7 **Períodos: 14**

UNIDAD 7	TITULO	Sesión
7.1	Arreglos Unidimensionales	34
7.1.1	Concepto y declaración de arreglos unidimensionales y ejercicios de arreglos sin funciones	35, 36
7.2	Paso de Arreglos a Funciones	37, 38
7.3	Ejercicios de aplicación (arreglos unidimensionales con funciones)	39, 40

UNIDAD 8 **Períodos: 16**

UNIDAD 8	TITULO	Sesión
8.1	Arreglos Bidimensionales (Proyecto para final del 2do parcial	41
8.1.1	Concepto y declaración de arreglos bidimensionales	42, 43
8.2	Paso de Arreglos a Funciones	44, 45
8.3	Ejercicios de aplicación (incluir también ejercicios con cadenas)	46, 47, 48

5. CONTENIDO PROGRAMÁTICO SYLLABUS

SESION	CONTENIDO	OBSERVACIONES
Sesión 1	Conceptos básicos de programación	
	Historia de Lenguaje C	
Sesión 2	Estructura General de un programa	
Sesión 3	Conceptos de librerías	
Sesión 4	Constantes, Concepto de Constantes	
	Declaración de Constantes	
Sesión 5	Variables	
	Concepto de Variables	
	Declaración de Variables	
	Asignación de valores a Variables	
Sesión 6	Expresiones Aritméticas	
	Operadores Aritméticos	
	Jerarquía de operadores aritméticos	
	Expresiones Relaciónales	
	Operadores Relaciónales	
	Jerarquía de operadores aritméticos y relacionales	
Sesión 7	Expresiones Booleanas	
	Operadores de verdad y tablas de verdad	
Sesión 8	Funciones de Entrada (scanf, cin, getch, gets, getchar, getc)	
Sesión 9	Funciones de Entrada (scanf, cin, getch, gets, getchar, getc)	
Sesión 10	Funciones de Salida (printf, cout, putc, putchar, gotoxy, cat)	
Sesión 11	Funciones de Salida (printf, cout, putc, putchar, gotoxy, cat)	
Sesión 12	Ejercicios de Aplicación	
Sesión 13	Sentencia If SIMPLE , IF Doble	
Sesión 14	IF Múltiple	
Sesión 15	IF Anidado	
Sesión 16	Sentencia Switch	
Sesión 17	Sentencia While	
Sesión 18	Sentencia Do/While (proyecto para final del primer parcial)	
Sesión 19	Sentencia For	
Sesión 20	Ejercicios de Aplicación	
Sesión 21	Ejercicios de Aplicación	
Sesión 22	Introducción a Funciones	
	Definición de una función	

	Llamada a una función	
	Declaración de una función	
Sesión 23	Funciones para manejo de números aleatorios	
Sesión 24	Envió de parámetros a una función	
Sesión 25	Parámetros por valor, Funciones para Manejo de Cadenas de Caracteres (strcpy, strcmp, strcmpi, strcat, strlen)	
Sesión 26	Funciones para Manejo de Cadenas de Caracteres (strcpy, strcmp, strcmpi, strcat, strlen)	
Sesión 27	Funciones para Manejo de Cadenas de Caracteres (strcpy, strcmp, strcmpi, strcat, strlen)	
Sesión 28	Ejercicios de Aplicación	
Sesión 29	Arreglos Unidimensionales	
Sesión 30	Concepto y declaración de arreglos unidimensionales y ejercicios de arreglos sin funciones	
Sesión 31	Concepto y declaración de arreglos unidimensionales y ejercicios de arreglos sin funciones	
Sesión 32	Paso de Arreglos a Funciones	
Sesión 33	Paso de Arreglos a Funciones	
Sesión 34	Ejercicios de aplicación (arreglos unidimensionales con funciones)	
Sesión 35	Ejercicios de aplicación (arreglos unidimensionales con funciones)	
Sesión 36	Concepto y declaración de arreglos bidimensionales	
Sesión 37	Concepto y declaración de arreglos bidimensionales	
Sesión 38	Paso de Arreglos a Funciones	
Sesión 39	Paso de Arreglos a Funciones	
Sesión 40	Ejercicios de aplicación (incluir también ejercicios con cadenas)	
Sesión 41	Ejercicios de aplicación (incluir también ejercicios con cadenas)	
Sesión 42	Ejercicios de aplicación (incluir también ejercicios con cadenas)	

6. METODOLOGIA

Metodología a utilizarse dentro del aula

1. Explicación de conceptos, planteamiento, análisis y resolución de problemas en el salón de clases.
2. Edición, ejecución, comprensión de problemas y dominio del contenido en el Laboratorio de Computación.
3. Talleres
4. Proyectos individuales y en grupos.
5. Discusión de temas de investigación, análisis de problemas, participación del estudiante.

7. EVALUACIÓN

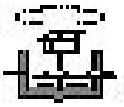
Criterio para la calificación de los trabajos.

- 30% Aportes, Lecciones, Deberes, Investigaciones, talleres, proyectos
- 70% Examen Escrito.

8. BIBLIOGRAFÍA BÁSICA

Nombre	Autor	Editorial	Edición	Básica ó Complementaria
Enciclopedia del lenguaje C	Francisco Ceballos			Complementaria
Programación en C y C++	Deitel && Deitel			Complementaria
Programación Estructurada	Luis Joyanez			Básica
Programación en C	Byron S. Gottfried	McGRAW-HILL		Básica
Internet				Complementaria

- Elaborado por: Ing. Marcia Bayas, Ing. David Benavides, Ing. Angel Ochoa



UNIVERSIDAD DE GUAYAQUIL
FACULTAD DE CIENCIAS MATEMATICAS Y FISICAS
CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES

PROGRAMA ANALITICO 2013-2014

1. DATOS GENERALES

ASIGNATURA: **PROGRAMACIÓN II**
CÓDIGO : **203** CRÉDITOS: **5**
PRE-REQUISITO: **PROGRAMACIÓN I**
PERÍODOS POR SEMANA: **6**
PERÍODOS POR SEMESTRE: **96 PERÍODOS (16 semanas)**

2. DESCRIPCIÓN SINTÉTICA

Asignatura que complementa la enseñanza de la programación Estructurada. Proporciona técnicas necesarias que permitirán al estudiante desarrollar software completos escritos con Lenguaje "C", además del manejo de las Estructuras de datos, almacenamientos en archivos y una introducción al manejo de hardware para comunicaciones de datos.

3. OBJETIVOS

Generales

- Analizar, diseñar e implementar cualquier programa estructurado en Lenguaje "C".
- Diferenciar de tipo de parámetros a funciones.
- Agrupar variables de diferentes tipos de datos.
- Almacenar datos en dispositivos.
- Utilizar memoria dinámica del computador.
- Identificar tipos y usos de las comunicaciones.

Específicos

- Manejar el concepto de recursividad.
- Utilizar direcciones de memoria a través de punteros.
- Invocar funciones con llamadas por referencia.
- Implementar relaciones entre punteros, arreglos y cadenas.
- Aplicar estructuras de datos compuestas y arreglos de estas.
- Transferir estructuras a funciones en llamadas por valor y en llamadas por referencia.
- Asignar y liberar memoria dinámicamente para objetos de datos.
- Manejar estructuras de datos enlazadas (Listas, Pilas y Colas), mediante el uso de punteros, estructuras autoreferenciadas y recursividad.
- Aplicar los archivos como medio de almacenamientos de datos a un dispositivo.
- Crear, leer, escribir, actualizar y desplazarse por los archivos secuénciales y aleatorios.
- Programar comunicaciones asincrónica y sincrónica.
- Implementar protocolos de comunicación.
- Programar los puertos seriales y paralelos.

4. CONTENIDOS PROGRAMÁTICOS

UNIDAD	NOMBRE
PRIMER PARCIAL	
1	FUNCIONES Y RECURSIVIDAD
2	PUNTEROS O APUNTADORES
3	ESTRUCTURAS Y UNIONES
SEGUNDO PARCIAL	
4	ASIGNACION DINAMICA DE MEMORIA
5	MANEJO DE ARCHIVOS Y DISPOSITIVOS
6	COMUNICACIÓN SERIAL Y PARALELOS DE DATOS.

5. CONTENIDO PROGRAMÁTICO SYLLABUS

UNIDAD 1

UNIDAD 1	FUNCIONES Y RECURSIVIDAD	SESIONES
1.1	Revisión de manejo de funciones	**1-2-3**
1.2	Definición de Recursividad	**3-4**
1.3	Algoritmos recursivos	**4**
1.4	Ejercicios de Aplicación	**4-5**

UNIDAD 2

UNIDAD 2	PUNTEROS O APUNTADORES	SESIONES
2.1	Apuntadores y direcciones	**6**
2.2	Apuntadores y arreglos unidimensionales, bidimensionales y multidimensionales	**7-8**
2.3	Aritmética de direcciones	**9**
2.4	Apuntadores a caracteres y funciones (llamadas por referencia)	**10**

UNIDAD 3

UNIDAD 3	ESTRUCTURAS Y UNIONES	SESIONES
3.1	Estructuras	**11**
3.1.1	Definición de Estructuras	**11**
3.1.2	Concepto de registros de datos	**11**
3.1.3	Declaraciones	**12-13**
3.1.4	Inicialización y Manejo	**14-15**
3.1.5	Como utilizar Estructuras con Funciones en llamadas por valor y llamadas por referencia.	**16-17**
3.1.6	Ejercicios de aplicación	**17-18**
3.2	Uniones	**19**
3.2.1	Definición de Uniones	**19**
3.2.2	Representación en memoria de Uniones	**19**
3.2.3	Declaraciones	**20**
3.2.4	Inicialización y Manejo	**20-21**
3.2.5	Ejercicios de aplicación	**20-21**

UNIDAD 4

UNIDAD 4	ASIGNACION DINAMICA DE MEMORIA	SESIONES
4.1	Funciones de asignación dinámica de memoria	**22**
4.2	Pilas Dinámicas	**22**
4.2.1	Concepto de Pila	**22**
4.2.2	Creación de una pila	**22-23**
4.2.3	Inserción, Eliminación, Recorrido y Búsqueda en una Pila	**22-23**
4.2.4	Manejo y ejercicios de aplicación de Pila	**23**
4.2	Colas Dinámicas	**24**
4.2.1	Concepto de Colas	**24**
4.2.2	Creación de una Cola	**24**
4.2.3	Inserción, Eliminación, Recorrido y Búsqueda en una Cola	**25-25**
4.2.4	Manejo y ejercicios de aplicación de Cola	**25**
4.3	Listas Dinámicas	**26**
4.3.1	Listas Simplemente enlazadas	**26**
4.3.1.1	Creación de una Lista	**26**
4.3.1.2	Inserción, Eliminación, Recorrido y Búsqueda en una Lista	**26-27**
4.3.1.3	Manejo y ejercicios de aplicación de Lista Simple	**27**
4.3.2	Listas Doblemente enlazadas	**28**
4.3.2.1	Creación de una Lista	**28**
4.3.2.2	Inserción, Eliminación, Recorrido y Búsqueda en una Lista	**28-29**
4.3.2.3	Manejo y ejercicios de aplicación de Lista Doble	**29**

UNIDAD 5

UNIDAD 5	MANEJO DE ARCHIVOS Y DISPOSITIVOS	SESIONES
5.1	Definiciones	**30**
5.2	Organización de Archivos	**30**
5.3	Tipos de Acceso	**30-31**
5.4	Apertura y Cierre de Archivos	**31**
5.5	Archivos tipo texto	**31**
5.5.1	Creación y Uso	**31-32**
5.5.2	Funciones: fputc/fgetc, fputs/fgets, fputw/fgetw	**32**
5.2.3	Ejercicios de Aplicación	**32**
5.6	Archivos tipo Binarios o archivos de registros	**33**
5.6.1	Creación y Uso	**33**
5.6.2	Funciones: fprintf/fscanf, fwrite/fread	**33-34**
5.6.3	Ejercicios de Aplicación	**34**
5.7	Acceso directo a datos	**35**
5.8	Ejercicios	**35**

UNIDAD 6

UNIDAD 6	COMUNICACIÓN SERIAL Y PARALELO DE DATOS	SESIONES
6.1	Definiciones	**36**
6.2	Comunicación asincrónica y sincrónica	**36**
6.3	La norma RS-32 para comunicación serial	**36-37**
6.4	Protocolos de comunicación	**37-38**
6.5	Conexión entre puertos seriales y MODEM	**39-40**
6.7	Programación de puertos seriales y paralelos	**40-41**
6.8	Ejemplos de transmisión de archivos.	**41-42**

6.- ESTRATEGIAS DE ENSEÑANZA-APRENDIZAJE

- Salón de clases
- Laboratorio de Computación.
- Investigaciones.

7.- EVALUACION

- 30% Aportes, Lecciones, Deberes, Investigaciones, Proyectos.
- 70% Examen.

8. BIBLIOGRAFÍA BASICA

N°	Titulo del autor	Titulo de la Obra	Editorial
1	Deitel && Deitel	Como programar en C/C++	
2	Francisco Ceballos	Enciclopedia del lenguaje C	
3	Luis Joyanez	Programación Estructurada en Lenguaje C	
4	Internet		

UNIVERSIDAD DE GUAYAQUIL
FACULTAD DE CIENCIAS MATEMATICAS Y FISICAS
CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES

PROGRAMA ANALITICO 2013-2014

1. DATOS GENERALES

ASIGNATURA: **PROGRAMACIÓN III**
CÓDIGO : **303** CRÉDITOS**: 5**
PRE-REQUISITO: **PROGRAMACIÓN II**
PERÍODOS POR SEMANA: **6**
PERÍODOS POR SEMESTRE: **96 PERÍODOS (16 semanas)**

2. DESCRIPCIÓN SINTÉTICA

El estudiante con los conocimientos adquiridos en programación I y programación estructurada, tiene las bases para complementar las diferentes técnicas de la programación, como lo es la Programación Orientada a Objetos.

3. OBJETIVOS

Generales

- Analizar, diseñar e implementar un programa orientado a objetos en el lenguaje C++.
- Aplicar las características que un lenguaje de programación orientado a objetos debe tener para su desarrollo.

Específicos

- El alumno debe tener claro los conceptos orientado a objetos y poder aplicarlos en otro lenguaje de programación.
- Aplicar sus conocimientos en el campo laboral.

4. CONTENIDOS PROGRAMÁTICOS

1 Introducción a la Programación Orientada a Objetos.
2 Clases y Objetos.
3 Herencia y Polimorfismo
4 Técnicas y Herramientas

5. CONTENIDO PROGRAMÁTICO SYLLABUS

SESIONES	CONTENIDO(TEMAS A TRATAR)	OBSERVACIONES
Sesión 1	1.1 Conceptos básicos de programación Orientada a objetos 1.2 Desarrollo del software: calidad, modularidad y reutilización	
Sesión 2	1.3 Modelamiento orientado a Objetos	
Sesión 3	1.3 Modelamiento orientado a Objetos	
Sesión 4	1.4 Programación orientada a Objetos	
Sesión 5	1.5 Revisión de la sintaxis básica orientada a Objetos	

Sesión 6	1.5 Revisión de la sintaxis básica orientada a Objetos	
Sesión 7	1.5 Revisión de la sintaxis básica orientada a Objetos	
Sesión 8	2.1 Clases y Objetos	
Sesión 9	2.1 Clases y Objetos	
Sesión 10	2.2 Atributos y métodos	
Sesión 11	2.2 Atributos y métodos	
Sesión 12	2.3 Paso de mensajes	
Sesión 13	2.3 Paso de mensajes	
Sesión 14	2.4 Tipo de acceso y formas de aplicarlo	
Sesión 15	2.4 Tipo de acceso y formas de aplicarlo	
Sesión 16	2.4 Tipo de acceso y formas de aplicarlo	
Sesión 17	2.5 Constructores y destructores	
Sesión 18	2.5 Constructores y destructores	
Sesión 19	2.6 Miembros especiales de una clase	
Sesión 20	2.7 Composición	
Sesión 21	2.8 Amistad 2.9 Sobrecarga de operadores	
Sesión 22	2.9 Sobrecarga de Operadores	
Sesión 23	3.1 Introducción a la herencia	
Sesión 24	3.2 Tipos de accesos para la herencia	
Sesión 25	3.2 Tipos de accesos para la herencia	
Sesión 26	3.3 Miembros protegidos	
Sesión 27	3.4 Sobreposición de métodos	
Sesión 28	3.5 Jerarquía de clases	
Sesión 29	3.5 Jerarquía de clases	
Sesión 30	3.6 Constructores y destructores	
Sesión 31	3.7 Amistad	
Sesión 32	3.8 Herencia múltiple	
Sesión 33	3.8 Herencia múltiple	
Sesión 34	3.9 Clases virtuales 3.10 Polimorfismo	

Sesión 35	3.11 Conversión de Objetos 3.12 funciones virtuales	
Sesión 36	3.13 Polimorfismo 3.14 Clases abstractas y concretas	
Sesión 37	3.15 Ejercicios aplicativos	
Sesión 38	3.15 Ejercicios aplicativos	
Sesión 39	3.15 Ejercicios aplicativos	
Sesión 40	4.1 Librería iostream. E/S de datos usando un lenguaje P.O.O.	
Sesión 41	4.2 E/S de archivos usando un lenguaje de P.O.O 4.3 Estructura de datos dinámicos usando clases.	
Sesión 42	4.4 Métodos de búsqueda y ordenamiento.	

6. METODOLOGÍA

Exposiciones

Trabajos en el laboratorio

Talleres prácticos en el laboratorio

Horas No Presenciales

Estos estudios independientes estarán comprendidos entre:

ESTUDIO INDEPENDIENTE	ESTIMADO DE PERÍODOS DEDICADOS
Lecturas de: texto, revistas, artículos, periódicos, etc. (mínimo 300 páginas)	20
Preparación de: Informes, presentaciones, ensayos, proyectos, investigaciones, casos	14
Estudio para: lecciones, aportes, exámenes (parcial, final), evaluaciones sobre lecturas, casos	10
Investigaciones: bibliográficas, de campo, Internet	10

7. EVALUACIÓN

- Aplicación de los conceptos orientados a objetos en los trabajos desarrollados.
- Explicación de los trabajos desarrollados.

8. BIBLIOGRAFÍA BÁSICA

Bibliografía Referencial

Nombre	Autor	Editorial	Edición	Básica ó Complementaria
Prog. Orientada a Objetos	**Luis Joyanes**			
P.O.O. con C++	**Fco. Javier Cevallos**			
Análisis y diseño orientado a Objetos en C++	**Fco. Javier Cevallos**			

Elaborado por: Ing. Bolivar Ramos Mosquera

UNIVERSIDAD DE GUAYAQUIL
FACULTAD DE CIENCIAS MATEMATICAS Y FISICAS
CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES

PROGRAMA ANALITICO 2013-2014

1. DATOS GENERALES

ASIGNATURA: **PROGRAMACIÓN ORIENTADA A OBJETOS**
CÓDIGO : **403** CRÉDITOS: 5
PRE-REQUISITO: **PROGRAMACIÓN III**
PERÍODOS POR SEMANA: **6**
PERÍODOS POR SEMESTRE: **96 PERÍODOS (16 semanas)**

2. DESCRIPCIÓN SINTÉTICA

Esta asignatura propiciará el análisis de los conceptos de la programación orientada a objetos como metodología de desarrollo de aplicaciones cuyo código de programación pueda ser reutilizado.

3. OBJETIVOS

Generales

- Definir los principales conceptos y elementos de la teoría orientada a objetos.
- Desarrollar aplicaciones con técnicas de orientación a objetos.

Específicos

- Manejar el lenguaje de programación JAVA, desde la creación de objetos hasta la programación en red.

4. CONTENIDOS PROGRÁMÁTICOS

CAPÍTULO 1 INTRODUCCION

1.1 Arquitectura para la ejecución de una Applet
1.2 Estructura de las Applet
1.3 Creación y Ejecución de una applets

CAPITULO 2 ACCESO A UNA BASE DE DATOS)

2.1 Base de Datos , Conectividad JDBC
2.2 Puente JDBC – ODBC
2.3 Manejo de la Información de una Base de Datos
2.4 Transacciones, Tipos de SQL en Java
2.5 Modelo relacional de Objetos y modelo de conexión

CAPÍTULO 3 GUI avanzadas en JAVA

3.1 Manejo de ambiente grafico
3.2 Manejo de GUI para el acceso a Archivos y BD
3.3 Menús, Ventanas y Cajas de Dialogo

CAPÍTULO 4 JAVA NETWORKING

4.1 Conceptos de Java NET
4.2 Conexión con URLS
4.3 Conexión con Sockets y Datagramas
4.4 Serializacion de Objetos
4.5 Comunicaciones basadas en HILOS

6. METODOLOGÍA

Horas Presenciales

Las clases serán interactivas, se trabajará en talleres y grupos en el laboratorio, habrán exposiciones sobre las investigaciones, evaluaciones sobre las lecturas y trabajos.

Horas No Presenciales

Los alumnos deberán dedicar 2 horas de estudios independientes (fuera de clase) por cada hora presencial que reciban.

Estos estudios independientes estarán comprendidos entre:

ESTUDIO INDEPENDIENTE	ESTIMADO DE HORAS DEDICADAS
Lecturas de: texto, revistas, artículos, periódicos, etc. (mínimo 50 páginas)	10 horas
Preparación de: Informes, presentaciones, ensayos, proyectos, investigaciones, casos	30 horas
Estudio para: lecciones, aportes, exámenes (parcial, final), evaluaciones sobre lecturas, casos	43 horas
Investigaciones: bibliográficas, de campo, internet	15 horas

7. EVALUACIÓN

Se evaluarán actividades como: tareas, lecturas, investigaciones, aportes, exposiciones y se tomarán tres exámenes.

Se calificará sobre 100 puntos (números enteros)

8. BIBLIOGRAFÍA BÁSICA

Nombre	Autor	Editorial	Edición	Básica ó Complementaria
PROGRAMACIÓN EN JAVA 2	**LUIS JOYANES AGUILAR**	McGraw Hill	Segunda [1996]	
COMO PROGRAMAR EN JAVA	**DEITEL & DEITEL**	PRENTICE HALL		
TUTORIAL DE JAVA 2	**AGUSTÍN FROUFE**	pagina WEB de Agustín Froufe	2do Manual	

Anexo No. 03

Misión, Visión y Perfil Profesional del Ingeniero en Sistemas Computacionales de la Universidad de Guayaquil

Misión

Formar profesionales en las ciencias informáticas, altamente calificados en el ámbito académico, científico, tecnológico, humanista y cultural, con sólidos valores éticos y morales; capaces de investigar e innovar para dar soluciones a los problemas técnicos y necesidades presentes y futuras del país.

Visión

Ser una carrera líder en la formación de ingenieros en sistemas computacionales comprometidos con la sociedad, que se proyectará como un conjunto de conocimientos, técnicas, procedimientos, metodologías y convenios para fomentar la investigación técnico – científica en la Tecnologías de la Información y Comunicaciones (TIC's), desarrollar habilidades profesionales entorno a la programación de soluciones informáticas que posibiliten la ingeniería y producción de software al servicio de otras áreas del conocimiento, y brindar respuesta a la demanda social de nuestra realidad ecuatoriana e internacional.

Perfil Profesional

El Ingeniero en Sistemas Computacionales graduado en la Universidad de Guayaquil es un profesional con sólidos conocimientos en las Ciencias de la Computación y las Tecnologías en Informática y estará capacitado para incursionar en las siguientes áreas:

Desarrollo de Tecnologías:

- Analizar, diseñar y programar sistemas informáticos
- Diseñar, programar y proveer tecnologías de mejoramiento de procesos dentro de las Organizaciones.
- Diseñar metodologías, planes de acción y programas informáticos para enfrentar problemas y/o retos no previstos a corto, mediano y largo plazo.
- Diseñar, programar y administrar redes de computadoras en diferentes plataformas tecnológicas.
- Diseñar, programar, implementar y evaluar sistemas digitales.
- Programar software utilitario y paquetes para computadoras de uso general.
- Proponer, asesorar y determinar criterios para la evaluación y control de alternativas de plataformas de Hardware y Software para satisfacer necesidades informáticas de una Organización.
- Incorporar los últimos avances de las Tecnologías de la Información y las Comunicaciones (TIC's) en la producción de software para la solución de problemas técnicos y de gestión empresarial.
- Desarrollar investigaciones científicas en el campo de las TIC's.

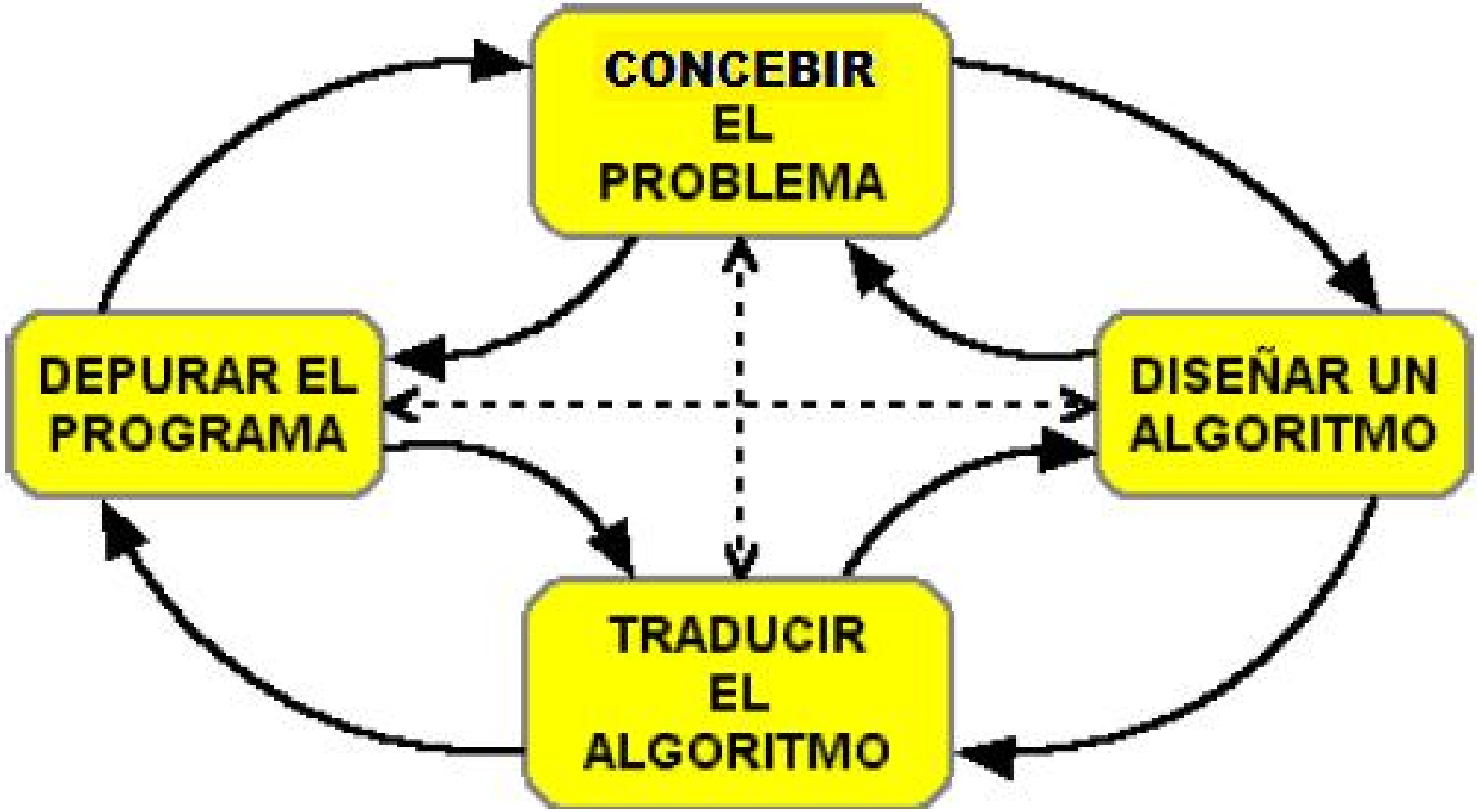
Administrativa:

- Diseñar, organizar, dirigir y administrar centros de cómputo y demás departamentos afines.
- Asesorar y auditar en proyectos y sistemas informáticos.
- Dirigir grupos interdisciplinarios para el desarrollo de sistemas informáticos.
- Asesorar y/o asistir a empresarios con tendencia a la automatización de empresas.
- Identificar, integrar y operar la infraestructura tecnológica computacional

Anexo No. 04

4 Invariantes del Proceso de Formación de la habilidad *programar*

(Fases para Elaborar un Programa de Computadora)



Anexo No. 05

Sistema de Tareas Docentes Integradoras para la formación de la habilidad *programar* en las asignaturas de programación de computadoras en los primeros años de la carrera de Ingeniería en Sistemas Computacionales

Tarea Integradora I: Concebir el problema a programar **(P1)**, **(P2)**, **(P3)**, **(PO).**

Objetivo: Analizar el dominio contextual del problema e identificar los datos de entrada, datos de salida, datos auxiliares, objetos, clases, atributos, condiciones, restricciones y la descripción de la solución del problema a resolverse por programación.

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

Acciones

- Identificar la incógnita central del problema a resolver por programación. **(P1)**, **(P2)**, **(P3)**, **(PO).**
- Analizar el dominio contextual del problema a programar. **(P1)**, **(P2)**, **(P3)**, **(PO).**
- Identificar las características esenciales de la información, variables y los datos presentes en el problema a programar. **(P1)**, **(P2)**, **(P3)**, **(PO)**
- Descomponer el problema en sus elementos principales variables y datos. **(P1)**, **(P2)**, **(P3)**.
- Identificar los datos de entrada, de salida, datos auxiliares del problema a resolver por programación. **(P1)**, **(P2)**, **(P3)**, **(PO).**
- Identificar en la situación problémica las diferentes condiciones tanto explícitas como implícitas, sobre los datos de entrada, datos auxiliares o de salida del problema a programar. **(P1)**, **(P2)**, **(P3)**, **(PO).**
- Describir el resultado esperado y las posibles alternativas de solución del problema. **(P1)**, **(P2)**, **(P3)**, **(PO).**

Operaciones

- Observar las características esenciales del problema. **(P1)**
- Describir la pertinencia del dominio científico/ tecnológico del problema. **(P1)**
- Clasificar los planteamientos propuestos como problemas y no problemas desde la viabilidad para resolverlos por programación. **(P1)**
- Elaborar un mapa conceptual de la clasificación de los problemas en función de la información que suministran: en estructurados y no estructurados **(P1), (P2)**.
- Ejemplificar problemas estructurados y no estructurados. **(P1), (P2)**.
- Comparar los datos del problema. **(P1), (P2)**, **(P3)**
- Identificar de los datos del problema cuales son constantes y cuáles son variables. **(P1), (P2),), (P3)**, **(PO).**
- Clasificar los tipos de variables en situaciones planteadas por el profesor. **(P1), (P2)**, **(P3)**, **(PO).**
- Interpretar cuáles son los datos de entrada y de salida en diferentes situaciones problémicas. **(P1), (P2)**, **(P3)**, **(PO).**

- Bosquejar un procedimiento en que los datos serán procesados para transformarlos en información. **(P1)**, **(P2)**, **(P3)**, **(PO).**
- Discriminar clases y objetos en un problema a programar **(P3)**, **(PO).**

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

- Deducir los datos auxiliares presentes en enunciados de problemas propuestos. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Argumentar el porqué de su caracterización como datos auxiliares.
- Crear un problema que tenga dos datos de entrada, uno constante y otro variable, no tenga datos auxiliares y tenga un solo dato de salida. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Identificar las condiciones explícitas sobre los datos y sus restricciones, en los enunciados de situaciones problémicas planteadas por el profesor. **(P1),** **(P2)**
- Deducir las condiciones implícitas sobre los datos presentes en los enunciados de los problemas planteados por el profesor. **(P1),** **(P2)**, **(P3)**
- Determinar las relaciones, operaciones y posibles estrategias de solución a partir de las variables y datos identificados, en las situaciones propuestas por el profesor. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Generalizar la secuencia de pasos para cumplir una tarea cotidiana de la vida real. **(P1),** **(P2)**, **(P3)**, **(PO).**

Tarea Integradora II: Diseñar la solución del problema a programar

Objetivo: Definir la estrategia de solución del problema a programar mediante el diseño de la lógica basada en algoritmos y modelos de la solución, desde los paradigmas de la programación estructurada y orientada a objetos

Acciones

- Comparar la estrategia de solución desde los paradigmas de la programación estructurada y la programación orientada a objetos. **(P1),** **(P2)**, **(P3)**, **(PO).**

Operaciones

- Secuenciar los pasos a seguir en algoritmos cotidianos de la vida real. **(P1)**

- Diseñar algoritmos desde la perspectiva Descendente (Top-Down) para el planteamiento de soluciones a problemas relacionados con las diferentes disciplinas del área. **(P1), (P2)**, **(P3)**.
- Representar gráficamente algoritmos mediante la construcción de Diagramas de Flujo como configuración de la estrategia de solución de problemas a programar. **(P1), (P2)**
- Representar algoritmos en lenguaje Pseudocódigo como configuración de la estrategia de solución de problemas a programar. **(P1), (P2)**
- Refinar algoritmos escritos en formato de diagramas de flujo y en estructura de pseudocódigo. **(P1), (P2)**, **(P3)**
- Definir con un enfoque de orientación a objetos, del problema a programar las especificaciones descritas en función de clases, objetos y atributos.
- Modelar las clases, objetos y sus relaciones del problema a programar, mediante el Lenguaje Unificado de Modelado (UML). **(P3)**, **(PO).**
- Diseñar el modelo de solución del problema a programar mediante todas las variantes del estándar UML
- Probar los algoritmos estructurados y los modelos de objetos, clases y relaciones. **(P3)**, **(PO).**

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

- Observar los diferentes tipos de aproximaciones y enfoques de la programación de computadoras en la actualidad. **(P1)**
- Comparar las ventajas y desventajas de la programación estructurada y la programación orientada a objetos. **(P3)**, **(PO).**
- Elaborar un mapa conceptual entre los distintos paradigmas de programación señalando sus ventajas y desventajas. **(P1), (P2)**, **(P3)**, **(PO).**
- Exponer las diferentes formas de representación de un algoritmo. **(P1), (P2)**, **(P3)**.
- Modelar el problema en pequeños módulos o subproblemas a partir la aplicación del diseño descendente (Top- Down). **(P1), (P2)**, **(P3)**.
- Elaborar un cuadro sinóptico que contenga los símbolos gráficos ANSI utilizados en flujogramas. **(P1), (P2)**.
- Desarrollar algoritmos a partir de problemas perfectamente delimitados, mediante diagramas de flujo. **(P1), (P2)**
- Ejemplificar que es un lenguaje de pseudocódigo en la construcción de algoritmos. **(P1), (P2)**, **(P3)**
- Presente y argumente la estructura de un algoritmo escrito en lenguaje pseudocódigo. **(P1), (P2)**, **(P3)**
- Construir un cuadro sinóptico de los diferentes tipos de datos y su clasificación. **(P1), (P2)**
- Argumentar en qué consisten las operaciones aritméticas, relacionales, lógicas y de carácter. **(P1), (P2)**.
- Ejemplificar los distintos tipos de instrucciones del lenguaje pseudocódigo: instrucciones de inicio/fin, de asignación, de lectura, de escritura y de bifurcación. **(P1), (P2)**, **(P3)**.

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

- Elaborar un cuadro de doble entrada para organizar los elementos básicos constitutivos de un programa: palabras reservadas, identificadores, caracteres especiales, constantes, variables, expresiones e instrucciones. **(P1), (P2), (P3), (PO).**
- Aplicar cambios en expresiones lógicas y algebraicas de un algoritmo modelo y analizar los resultados obtenidos. **(P1), (P2), (P3), (PO).**
- Describir en el aula a partir de un lenguaje de pseudocódigo, las caracteríticas y propiedades de los elementos de programación: Bucles, contadores, acumuladores, interruptores y estructuras. **(P1), (P2), (P3)**
- Construir en clase algoritmos que incluyan bucles, contadores, acumuladores, interruptores y estructuras. **(P1), (P2), (P3)**.
- Ejemplificar las estructuras secuenciales, selectivas y repetitivas, desde un lenguaje de pseudocódigo. **(P1), (P2), (P3)**.
- Analizar la sintaxis del lenguaje de pseudocódigo en cada una de las instrucciones y documente las operaciones lógicas y de control. **(P1), (P2), (P3), (PO).**
- Crear algoritmos donde se utilicen estructuras de flujo de control tanto de repetición como de selección. **(P1), (P2), (P3), (PO).**
- Modelar algoritmos a partir de problemas perfectamente delimitados, mediante Pseudocódigo, indicando las decisiones de diseño adoptadas y las decripciones del significado de cada línea de instrucciones. **(P1), (P2), (P3), (PO).**
- Diseñar algoritmos que utilicen constantes, variables y diferentes tipo de datos. **(P1), (P2), (P3)**.
- Representar algoritmos que empleen operadores aritméticos, lógicos y relacionales. **(P1), (P2), (P3), (PO).**
- Crear algoritmos que incluyan en su estructura control de flujos y bucles.

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

(P1), (P2), (P3), (PO).

- Innovar algoritmos que implementen métodos y funciones. **(P1), (P2), (P3), (PO).**
- Elaborar un reporte de los conceptos de estructuras de datos en programación de computadoras. **(P1), (P2), (P3).**
- Debatir en clase acerca de las estructuras de datos estáticas. **(P1), (P2), (P3), (PO).**
- Exponer las operaciones que se pueden realizar con cadenas de caracteres. **(P1), (P2).**
- Elaborar un mapa conceptual de los diferentes tipos de arreglos y socializarlos en clase. **(P1), (P2).**
- Crear algoritmos que utilicen archivos y cadenas. **(P1), (P2), (P3), (PO).**
- Construir algoritmos que utilicen arreglos unidimensionales y multidimensionales. **(P1), (P2), (P3), (PO).**
- Diseñar algoritmos para implementar las operaciones básicas en arreglos. **(P1), (P2), (P3).**
- Explique la clasificación de archivos: binarios y de texto.**(P2).**
- Diseñar algoritmos para implementar las operaciones básicas en archivos.
- Elaborar una tabla con sus definiciones de las estructuras de datos dinámicas: listas, pilas, colas, listas enlazadas, árboles y grafos. **(P2), (P3), (PO).**
- Determinar los métodos básicos en el manejo de listas, pila, colas, árboles y grafos. **(P2), (P3), (PO).**
- Describir los métodos de búsqueda, ordenamiento y recursión en las estructuras dinámicas: listas, pilas, colas, listas enlazadas, árboles y grafos. **(P2), (P3), (PO).**
- Diseñar algoritmos de búqueda y ordenamiento basados en recursión. **(P2), (P3), (PO).**
- Aplicar los distintos tipos de datos, operadores, instrucciones y estructuras, de control, estructura de datos estáticas y dinámicas, desarrollando su habilidad matemática al manejar la lógica computacional. **(P2), (P3).**

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

- Depurar algoritmos mediante la aplicación de Pruebas de Escritorio realizando conclusiones pertinentes a cada propuesta. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Valorar la ejecución manual y detecte errores en las instrucciones lógicas y de control de flujo. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Seleccionar un objeto cotidiano de la vida real y describa sus características.
- Crear una definición de los conceptos de objetos, clases, atributos de forma simple y entendible. **(PO).**
- Debatir en clase las definiciones de objetos, clases y atributos de otros compañeros, para enriquecer la propia y consensuar una definición grupal. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Describir las clases, los objetos y sus relaciones desde la concepción del problema a programar. **(P3)**, **(PO).**
- Investigar los conceptos principales del paradigma de programación orientado a objetos. **(P3)**, **(PO).**
- Identificar ejemplos de la vida real que apliquen o manifiesten dichos conceptos. **(PO).**
- Sintetizar una definición propia de objetos de forma simple y entendible.
- Debatir en clase las definiciones de objetos y clases de otros compañeros para enriquecer la propia y consensar una grupal. **(PO).**
- Argumentar la información que requiere el lenguaje UML referente al modelado de clases. **(P2)**, **(P3)**, **(PO).**
- Diseñar diagramas de clases aplicados a distintos problemas a programar. **(P2)**, **(P3)**, **(PO).**
- Diseñe diagramas de casos de uso aplicados a distintos problemas a programar. **(P2)**, **(P3)**, **(PO).**
- Diseñe diagramas de estado aplicados a distintos problemas a programar. **(P2)**, **(P3)**, **(PO).**
- Diseñe diagramas de cajas aplicados a distintos problemas a programar**(P3)**, **(PO).**.
- Emplee un software para probar el diagrama de flujo diseñado. **(P1),** **(P2)**, **(P3)**, **(PO).**

- Clasifique los errores de lógica y los errores de sintaxis. **(P1),** **(P2)**, **(P3)**, **(PO).**

Tarea Integradora III: Traducir a un lenguaje de programación, la solución del problema a programar **(P1),** **(P2)**, **(P3)**, **(PO).**

Objetivo: Codificar el algoritmo y/o el modelo orientado a objetos UML mediante un lenguaje de programación para la solución del problema a programar

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

Acciones

- Codificar en las suite de desarrollo de los lenguajes de programación más usados en el mercado. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Utilizar el diccionario de comandos de los lenguajes de programación de mayor uso en el mercado. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Aplicar las reglas sintácticas en la codificación de los algoritmos en el lenguaje de programación. **(P2)**, **(P3)**, **(PO).**
- Aplicar las palabras reservadas, librerías de funciones, APIS de los lenguajes de programación de mayor uso en el mercado. **(P3)**, **(PO).**
- Documentar las instrucciones o sentencias del programa. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Ejecutar el proceso de compilación del código fuente del programa escrito. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Ejecutar el proceso de depuración de errores lógicos y de sintaxis del programa. **(P1),** **(P2)**, **(P3)**, **(PO).**

Operaciones

- Argumentar las diferencias entre lenguajes compilados, interpretados, estructurados y orientados a objetos para seleccionar el lenguaje y las herramientas adecuadas. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Evaluar el lenguaje de programación idóneo para la implementación del algoritmo diseñado. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Valorar estadísticamente los lenguajes de programación de mayor uso en el mundo. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Comparar y clasificar los lenguajes de programación de ambiente de desarrollo tipo consola caracter y los de ambiente visual, de mayor uso en el mundo. **(P1),** **(P2)**, **(P3)**
- Sintetizar el diccionario de comandos, palabras reservadas y funciones básicas, a partir de la revisión de los manuales de usuario de los dos lenguajes de programación de mayor uso en el mundo. **(P1),** **(P2)**, **(P3)**
- Aplicar las reglas de sintaxis en la escritura de las instrucciones y sentencias de un lenguaje de programación. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Describir los tipos de datos, operadores y expresiones que manejan los lenguajes de programación sugeridos por el profesor. **(P1),** **(P2)**, **(P3)**, **(PO).**

- Implementar la metodología de Desarrollo Rápido de Aplicaciones RAD. **(P2)**, **(P3)**, **(PO).**

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

- Crear estructuras sintácticas que maneje el lenguaje de programación para las instrucciones de decisión, repetición. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Innovar un algoritmo escrito en lenguaje pseudocódigo que maneja expresiones y estructuras de control, tradúzcalo a dos lenguajes de programación recomendados por el profesor. **(P1),** **(P2)**, **(P3)**.
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo representado en diagrama de flujo que incluye la declaración de funciones y paso de parámetros. **(P1),** **(P2)**
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocodigo que incluye la declaración e inicialización de arreglos unidimensionales. **(P1),** **(P2).**
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocodigo que incluye la declaración e inicialización de arreglos bidimensionales o matrices. **(P1),** **(P2)**, **(P3)**.
- Codificar en un lenguaje de programación sugerido por el profesor, un algoritmo representado en diagrama de flujo que incluye la declaración e inicialización de cadenas y funciones para manejo de cadenas de caracteres. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la creación de una estructura. **(P1),** **(P2)**, **(P3)**
- Codificar en un lenguaje de programación sugerido por el profesor, un algoritmo representado en diagrama de flujo que incluye la declaración y manejo de punteros. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación de una clase. **(P2)**, **(P3)**, **(PO).**

- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación de funciones miembros de una clase. **(P3)**, **(PO).**
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación de constructores y destructores de una clase. **(P3)**, **(PO).**
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación de funciones, operadores y flujos de datos sobrecargados. **(P2)**, **(P3)**, **(PO).**
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación de manejo de memoria dinámica. **(P2)**, **(P3)**, **(PO).**
- Escribir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación del concepto de listas encadenadas: adición y eliminación de nodos. **(P1),** **(P2)**, **(P3)**
- Codificar en un lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación del recorido de listas encadenadas. **(P2)**, **(P3)**
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación de métodos de ordenamiento.**(P2)**, **(P3)**, **(PO).**
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación de métodos de ordenamiento. **(P2)**, **(P3)**, **(PO).**
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación del concepto de listas doblemente encadenadas: adición y eliminación de nodos. **(P2)**, **(P3)**, **(PO).**

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación del recorrido de listas doblemente encadenadas. **(P2)**, **(P3)**.
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación de métodos de ordenamiento en listas doblemente encadenadas. **(P2)**, **(P3)**.
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación del concepto de listas circulares simples: adición y eliminación de nodos. **(P2)**, **(P3)**.
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación del recorrido de listas circulares simples. **(P2)**, **(P3)**.
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación del concepto de listas circulares dobles: adición y eliminación de nodos. **(P2)**, **(P3)**.
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación del recorrido de listas circulares dobles.**(P2)**, **(P3)**.
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación de pilas con medios d almacenamiento estática. **(P2)**, **(P3)**, **(PO)**.
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación de colas con medios d almacenamiento estática. **(P2)**, **(P3)**, **(PO)**.
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación de pilas con medios d almacenamiento dinámica. **(P2)**, **(P3)**,

(PO).

- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación de colas con medios d almacenamiento dinámica. **(P2)**, **(P3)**, **(PO).**
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación del manejo y operación de pilas. **(P2)**, **(P3)**, **(PO).**
- Traducir al lenguaje de programación sugerido por el profesor, un algoritmo escrito en pseudocódigo que incluye la implementación del manejo y operación de colas. **(P2)**, **(P3)**, **(PO).**
- Investigar en fuentes de información los conceptos y reglas para implementar clases abstractas en un lenguaje de programación orientado a objetos.**(P3)**, **(PO).**
- Crear clases abstractas en clases base que no requieran ser instanciadas con al menos un método abstracto para que sea implementado por sus clases derivadas en múltiples formas. **(PO).**
- Implementar una clase con todos sus comportamientos abstractos. **(PO).**
- Analizar en diversas fuentes de información el concepto de interfaz y compararlo con la clase cien por ciento abstracta. **(P3)**, **(PO).**
- Implementar interfaces para definir los comportamientos que una clase deberá de tener al implementarla. **(PO).**
- Construir interfaces para definir los comportamientos que una clase deberá de tener al implementarla. **(P3)**, **(PO).**
- Crear una misma interfaz en diferentes clases para dar en cada una un comportamiento diferente a sus métodos. **(P3)**, **(PO).**
- Aplicar herencia de interfaces para especializar los comportamientos que las clases podrán implementar.**(PO).**

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

- Asignar variables miembro de tipo clase abstracta o interfaz para que en tiempo de ejecución se inicialice con diferentes subtipos o implementaciones de las mismas, y se demuestre así, toda la flexibilidad del polimorfismo al cambiar el comportamiento de un objeto en tiempo de ejecución. **(PO).**
- Crear un programa que deliberadamente genere excepciones comunes para identificar: sus nombres, sus causas, su comportamiento, y reporte de error. **(PO).**
- Innovar una clase con varios métodos invocándose en cadena, donde el último método genere una excepción para estudiar y comprender la propagación de las mismas. **(PO).**
- Implemente la estructura selectiva para atrapar excepciones de diferentes tipos, y prevenir la interrupción de ejecución de un programa. **(P3)**, **(PO).**
- Analizar situaciones en las que un método no pueda devolver un valor de retorno como indicador de un error interno, y tenga la necesidad de levantar una excepción por el usuario que le indique que su función no pudo ser realizada. **(P3)**, **(PO).**
- Codificar la instanciación y lanzamiento de excepciones definidas por el lenguaje para situaciones en que no es posible regresar un valor desde un método que indique una condición de error interno. **(P3)**, **(PO).**
- Discriminar condiciones de error requeridas por el usuario y no previstas por el lenguaje que requieran la creación de un nuevo tipo de excepción. **(P3)**, **(PO).**
- Crear un nuevo tipo de excepción definido por el usuario heredando de la clase base de las excepciones o alguna otra ya definida por el lenguaje que más se aproxime al comportamiento deseado del usuario. **(PO).**
- Experimentar el lanzamiento, propagación y manejo de una excepción definida por el usuario. **(PO).**

- Indagar en fuentes de información los conceptos y metodologías para manipular archivos de texto y binarios en un lenguaje de programación orientado a objetos. **(P3)**, **(PO).**
- Observar una clase que cree, consulte, modifique y borre archivos de texto.**(P2)**, **(P3)**, **(PO).**
- Implemente una clase que cree, consulte, modifique y borre archivos binarios. **(P2)**, **(P3)**, **(PO).**
- Crear un caso de estudio que requiera el uso de archivos para que sea resuelto por el alumno. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Describir una Interfaz Gráfica GUI y sus componentes. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Elaborar un cuadro sinóptico de las jerarquías de clases para las GUI. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Construir la interfaz gráfica de un sistema propuesto por el profesor. **(P3)**, **(PO).**
- Diseñar clases con atributos públicos para exponer y comprender la vulnerabilidad de los datos. **(P2)**, **(P3)**, **(PO).**
- Demostrar la protección de los atributos con modificadores de acceso privados o protegidos **(P3)**, **(PO).**
- Implementar métodos públicos para otorgar acceso seguro a los mismos. **(P3)**, **(PO).**
- Argumentar como se debe reunir dentro de una clase los miembros necesarios para resolver un problema en particular, y así implementar el encapsulamiento. **(P2)**, **(P3)**, **(PO).**
- Instanciar objetos para identificar el nacimiento y muerte de los mismos. **(P3)**, **(PO).**
- Diseñar constructores y destructores para las clases, de manera que permitan dar un valor inicial a sus atributos cuando nazcan sus objetos, o liberar recursos cuando mueran los mismos. **(P3)**, **(PO).**
- Identificar los comportamientos de una clase que pueden variar dependiendo del paso, cantidad, tipo u orden de

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

argumentos. **(P3)**, **(PO).**

- Implementar cada variación del comportamiento en métodos sobrecargados para agregar flexibilidad a la clase. **(P3)**, **(PO).**
- Identificar operaciones que puedan ser realizadas entre dos objetos de la misma clase. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Seleccionar un operador existente del lenguaje y sobrecargarlo en la clase de los objetos para implementarles dicha funcionalidad. **(P2)**, **(P3)**, **(PO).**
- Analizar las analogías taxonómicas de los seres vivos que compartan rasgos comunes por estar relacionados mediante una herencia genética e identificar la especie a la que pertenecen. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Identificar los atributos y comportamientos propios de una especie que comparten los animales pertenecientes a ella. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Analizar objetos reales que compartan características comunes por pertenecer a una misma categoría de objetos. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Determinar los atributos y comportamientos propios de una categoría de objetos que compartan todos sus miembros. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Analizar la propiedad de herencia y su implementación en un lenguaje de programación orientado a objetos. **(P2)**, **(P3)**, **(PO).**
- Crear una clase base para una especie de animales con los atributos y comportamientos comunes a todos los animales pertenecientes a ella. **(P2)**, **(P3)**, **(PO).**
- Implementar clases derivadas para animales pertenecientes a la misma especie de la cual se programó la clase base anteriormente. **(P2)**, **(P3)**, **(PO).**
- Especializar cada clase derivada con comportamientos y atributos específicos de un tipo de animal para identificarlo y distinguirlo de los demás. **(P3)**, **(PO).**

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

- Crear varias instancias de clases derivadas diferentes para verificar la existencia de los miembros heredados comunes en todas ellas, y la diversidad de sus especializaciones.**(P2)**, **(P3)**, **(PO).**

- Implementar la sobrecarga en los constructores de las clases base y derivadas para analizar y experimentar el comportamiento y uso de los constructores en combinación con la herencia. **(P3)**, **(PO).**
- Comparar qué animales u objetos de la vida real rompen algún comportamiento heredado para reinventar el suyo propio por sobre el resto de sus parientes que siguen respetando el heredado. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Crear un método en una clase derivada para sobrescribir el de su clase base e introducirse al polimorfismo. **(P2)**, **(P3)**, **(PO).**
- Determinar clases base que no requieran ser instanciadas, o que carezcan de sentido para ello por ser abstractas. **(P2)**, **(P3)**, **(PO).**
- Consultar bibliográficamente y relacionar los componentes de la Metodología RAD. **(P1),** **(P2)**, **(P3)**, **(PO).**
- Generar automáticamente los formularios dentro de los proyectos de programación. **(PO).**
- Generar automáticamente los reportes dentro de los proyectos de programación. **(PO).**
- Crear con la herramienta RAD la definición automática de clases y objetos de la aplicación. **(P2)**, **(P3)**, **(PO).**
- Codificar mediante la herramienta RAD los objetos en la definición de la lógica de negocios de una aplicación. **(P3)**, **(PO).**
- Compilar los programas fuentes en el compilador instalado en las pc que recomienda el profesor. **(P1),** **(P2)**, **(P3)**, **(PO).**

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

- Documentar las instrucciones de su programa con una descripción sencilla de la acción que se está ejecutando. **(P1), (P2), (P3), (PO).**

Tarea Integradora IV: Depurar el programa

Objetivo: Perfeccionar el programa escrito en un lenguaje de programación a fin de que esté listo para la puesta en marcha. **(P1), (P2), (P3), (PO).**

Acciones

- Probar la ejecución del programa mediante de técnicas de escritorio. **(P1), (P2), (P3), (PO).**
- Probar la ejecución del programa mediante de técnicas de datos de prueba. **(P1), (P2), (P3), (PO).**
- Manejar el modo de depuración del programa asistido por la suite de desarrollo. **(P1), (P2), (P3), (PO).**
- Ejecutar los programas escritos en un lenguaje de programación. **(P1), (P2), (P3), (PO).**

(P1): Programación I

(P2): Programación II

(P3): Programación III

(PO): Programación Orientada a Objetos

Operaciones

- Compilar programas recomendados por el profesor. **(P1), (P2), (P3), (PO).**
- Ejecutar programas recomendados por el profesor, en modo de depuración disponible en la suite de desarrollo recomendada por el profesor. **(P1), (P2), (P3), (PO).**
- Determinar los errores lógicos en los programas sugeridos por el profesor. **(P1), (P2), (P3), (PO).**
- Detectar los errores de sintaxis en los programas sugeridos por el profesor. **(P1), (P2), (P3), (PO).**
- Optimizar el código en los programas que les facilitará el profesor. **(P1), (P2), (P3), (PO).**
- Comprobar con técnicas de escritorio los programas recomendados por el profesor. **(P1), (P2), (P3), (PO).**
- Exponer los resultados de la ejecución de programas. **(P1), (P2), (P3), (PO).**
- Monitorear la ejecución de un programa en modo debug **(P1), (P2), (P3), (PO).**
- Secuenciar los pasos para compilar un programa. **(P1), (P2), (P3), (PO).**
- Exponer un cuadro sinóptico el procedimiento de ejecución de un programa en el lenguaje de programación o suite de desarrollo recomendada por el profesor. **(P1), (P2), (P3), (PO).**

Anexo No. 06

Comunicación a Expertos

Presentación:

En la Universidad de Guayaquil se desarrolla una investigación que aborda la temática de la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación de computadoras en los primeros años de la Carrera de Ingeniería en Sistemas Computacionales. En la tesis se presenta un Sistema de Tareas Docentes Integradoras, cuya aplicación propicie la formación de esta habilidad.

Para la concreción de tal propósito, solicitamos gentilmente que usted nos apruebe su conformidad de estar en condiciones para ofrecer sus criterios en calidad de Experto y poder validar el Sistema de Tareas Docentes Integradoras, que se propone.

Marque con (X): SI _____ NO ____

Si su respuesta es positiva favor de llenar los siguientes datos:

Nombres y apellidos:_____

Marque con (X): Docente Contratado: _____ Docente con Nombramiento: _____

Más Alta Categoría Académica: Pregrado:___ Máster:____Doctor:____

Institución donde labora:_____

Dirección del centro:_____

Teléfono del centro:_____

Dirección particular:_____

Teléfono Convencional:_____Móvil: _____ Email:_____

Encuesta a Expertos

Para la determinación del Coeficiente de Competencia

Presentación:

Teniendo en cuenta su disposición a cooperar en calidad de posible experto, se pone a su valoración los criterios expuestos en las dos tablas siguientes con el objetivo de valorar el **coeficiente de conocimiento** y el **coeficiente de argumentación** sobre el Sistema de Tareas Docentes Integradoras que se propone para la formación de la habilidad *programar* en los estudiantes de los primeros años de la carrera de Ingeniería en Sistemas Computacionales. Le agradecemos de antemano su colaboración:

Cuestionario:

1. Marque con una X en escala creciente del 1 al 10 el grado de conocimiento o información sobre el tema abordado:

   1___ 2___ 3___ 4___ 5___ 6___ 7___ 8___ 9___ 10___

2. Valore los aspectos que influyen sobre el nivel de argumentación o fundamentación que usted posee sobre el tema objeto de estudio. Marque con X según corresponda, atendiendo a las fuentes de argumentación que se le sugieren.

FUENTES DE ARGUMENTACIÓN	ALTO	MEDIO	BAJO
Experiencia teórica vinculada a la temática que se aborda.			
Bibliografía nacional consultada sobre la temática.			
Bibliografía internacional consultada sobre el tema.			
Estudios e investigaciones realizados en relación con la evaluación de profesores universitarios, concretados de manera particular en una metodología.			
Su intuición.			

Anexo No. 07

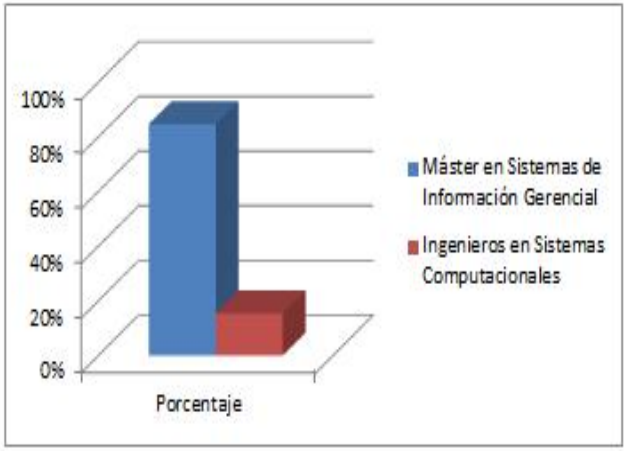
Tabla Patrón

FUENTES DE ARGUMENTACIÓN		ESCALA DE VALORACIÓN		
		ALTO	MEDIO	BAJO
1	Su experiencia teórica	0.30	0.20	0.10
2	Su experiencia práctica	0.50	0.40	0.20
3	Bibliografía Nacional Consultada	0.05	0.05	0.05
4	Bibliografía Internacional Consultada	0.05	0.05	0.05
5	Su conocimiento del Estado del Problema	0.05	0.05	0.05
6	Su intuición	0.05	0.05	0.05

Anexo No. 08

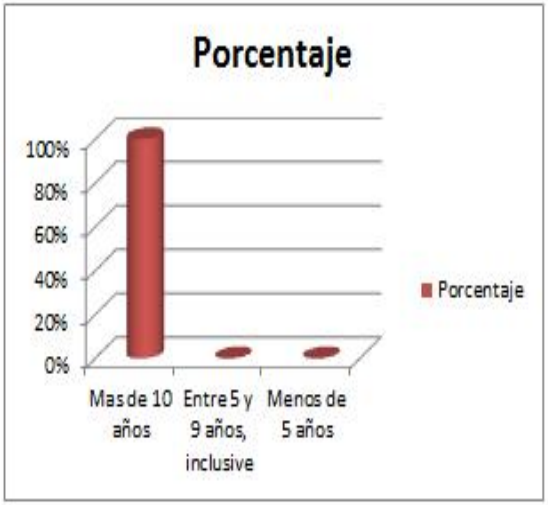
Estadística y Caracterización de Expertos de la Universidad de Guayaquil

Formación Académica	Porcentaje
Máster en Sistemas de Información Gerencial	85%
Ingenieros en Sistemas Computacionales	15%
Total	100%



Relación Contractual con la U. de Guayaquil	Cantidad
Profesor Titular a Tiempo Completo	8
Profesor Contratado a Tiempo Parcial	5
Total	13

Docencia en Programación de Computadoras	Porcentaje
Mas de 10 años	100%
Entre 5 y 9 años, inclusive	0%
Menos de 5 años	0%
Total	100%



Anexo No. 09

Guía para Orientar la Valoración de los Expertos del

Sistema De Tareas Docentes Integradoras

Estimado colega considerando su preparación y Coeficiente de Competencia en el tema, usted ha sido seleccionado para realizar una valoración del Sistema de Tareas Docentes Integradoras para la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación en los primeros años de la carrera de Ingeniería en Sistemas Computacionales. A tal efecto, se le facilita la información que se ha considerado necesaria para que realice la valoración correspondiente.

Se necesita que asuma la tarea con la responsabilidad que requiere este trabajo y agradecemos su valiosa colaboración.

En la tabla que se muestra a continuación le proponemos los criterios sobre los cuales nos interesaría conocer sus valoraciones.

VALORACIÓN	DESCRIPCIÓN DE LA VALORACIÓN
Muy Adecuado (MA)	Se considera aquel aspecto que es óptimo, en el cual se expresan todas y cada una de las propiedades, consideradas como componentes esenciales para determinar la calidad del objeto que se evalúa.
Bastante Adecuado (BA)	Se considera aquel aspecto que expresa en casi toda su generalidad las cualidades esenciales del objeto que se evalúa, siendo capaz de representar con un grado bastante elevado, los rasgos fundamentales que tipifican su calidad.
Adecuado (A)	Se considera aquel aspecto que tiene en cuenta una parte importante de las cualidades del objeto a evaluar, las cuales expresan elementos de valor con determinado nivel de suficiencia, aunque puede ser susceptible de perfeccionamiento en situaciones poco significativas.
Poco Adecuado (PA)	Se considera aquel aspecto que expresa un bajo nivel de adecuación en relación con el estado deseado del objeto que se evalúa al expresarse carencias en determinados componentes, considerados esenciales para determinar su calidad.
Inadecuado (I)	Se considera aquel aspecto en el que se expresan marcadas limitaciones y contradicciones que no le permiten adecuarse a las cualidades esenciales que determinan la calidad del objeto que se evalúa por lo que no resulta procedente.

Aspectos a Valorar en relación con el Sistema de Tareas Docentes Integradoras	ESCALA VALORATIVA				
	MA	BA	A	PA	I
[1]Sistematicidad de las cuatro invariantes funcionales para la formación de la habilidad programar: Concebir el problema, Diseñar el Algoritmo Solución; Traducir lo a un lenguaje de programación y Depurarlo.					
[2]Pertinencia de las etapas didácticas declaradas en el Sistema de Tareas Docentes Integradoras					
[3]Aplicabilidad del Sistema de Tareas Docentes Integradoras en el contexto de las Carreras de Ingeniería en Sistemas Computacionales de las universidades ecuatorianas					
[4]El Sistema de Tareas Docentes Integradoras propicia la participación y el protagonismo de los estudiantes en el proceso.					
[5]Pertinencia de los acciones y operaciones que se proponen en las tareas.					
[6]Contribución a la formación de la habilidad *programar* desde la integración sistémica e interrelacionada de los contenidos esenciales de las asignaturas de programación en los primeros años de la carrera.					
[7]El Sistema de Tareas Docentes Integradoras propicia en los estudiantes la transformación del desempeño en la habilidad *programar* a partir de los modos de actuación del Ingeniero de Sistemas.					

[8]Flexibilidad del Sistema de Tareas Docentes Integradoras para la dosificación y tratamiento individual de los niveles de asimilación de contenidos de los estudiantes de programación.					
[9]Asequibilidad del Sistema de Tareas para la evaluación de la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación en los primeros años de su carrera.					
[10]Posibilidad de generalización del Sistema de Tareas Docentes Integradoras para la formación de habilidades profesionales en los estudiantes de Ingeniería en Sistemas Computacionales.					

Anexo No. 10
Tabla de Frecuencias Observadas

Aspectos a Valorar en relación con el Sistema de Tareas Docentes Integradoras	ESCALA VALORATIVA				
	MA	BA	A	PA	I
[1]Sistematicidad de las cuatro invariantes funcionales para la formación de la habilidad programar: Concebir el problema, Diseñar el Algoritmo Solución; Traducir lo a un lenguaje de programación y Depurarlo.	10	2	1	0	0
[2]Pertinencia de las etapas didácticas declaradas en el Sistema de Tareas Docentes Integradoras	9	2	2	0	0
[3]Aplicabilidad del Sistema de Tareas Docentes Integradoras en el contexto de las Carreras de Ingeniería en Sistemas Computacionales de las universidades ecuatorianas	9	3	1	0	0
[4]El Sistema de Tareas Docentes Integradoras propicia la participación y el protagonismo de los estudiantes en el proceso.	7	3	3	0	0
[5]Pertinencia de los acciones y operaciones que se proponen en las tareas.	8	5	0	0	0
[6]Contribución a la formación de la habilidad *programar* desde la integración sistémica e interrelacionada de los contenidos esenciales de las asignaturas de programación en los primeros años de la carrera.	10	1	2	0	0
[7]El Sistema de Tareas Docentes Integradoras propicia en los estudiantes la transformación del desempeño en la habilidad *programar* a partir de los modos de actuación del Ingeniero de Sistemas.	10	1	2	0	0
[8]Flexibilidad del Sistema de Tareas Docentes Integradoras para la dosificación y tratamiento individual de los niveles de asimilación de contenidos de los estudiantes de programación.	6	7	0	0	0
[9]Asequibilidad del Sistema de Tareas para la evaluación de la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación en los primeros años de su carrera.	7	4	2	0	0
[10]Posibilidad de generalización del Sistema de Tareas Docentes Integradoras para la formación de habilidades profesionales en los estudiantes de Ingeniería en Sistemas Computacionales.	8	1	4	0	0

Anexo No. 11
Tabla de Frecuencias Acumuladas

Aspectos a Valorar en relación con el Sistema de Tareas Docentes Integradoras	ESCALA VALORATIVA				
	MA	BA	A	PA	I
[1]Sistematicidad de las cuatro invariantes funcionales para la formación de la habilidad programar: Concebir el problema, Diseñar el Algoritmo Solución; Traducir lo a un lenguaje de programación y Depurarlo.	13	3	1	0	0
[2]Pertinencia de las etapas didácticas declaradas en el Sistema de Tareas Docentes Integradoras	13	4	2	0	0
[3]Aplicabilidad del Sistema de Tareas Docentes Integradoras en el contexto de las Carreras de Ingeniería en Sistemas Computacionales de las universidades ecuatorianas	13	4	1	0	0
[4]El Sistema de Tareas Docentes Integradoras propicia la participación y el protagonismo de los estudiantes en el proceso.	13	3	6	0	0
[5]Pertinencia de los acciones y operaciones que se proponen en las tareas.	13	5	0	0	0
[6]Contribución a la formación de la habilidad *programar* desde la integración sistémica e interrelacionada de los contenidos esenciales de las asignaturas de programación en los primeros años de la carrera.	13	3	2	0	0
[7]El Sistema de Tareas Docentes Integradoras propicia en los estudiantes la transformación del desempeño en la habilidad *programar* a partir de los modos de actuación del Ingeniero de Sistemas.	13	3	2	0	0
[8]Flexibilidad del Sistema de Tareas Docentes Integradoras para la dosificación y tratamiento individual de los niveles de asimilación de contenidos de los estudiantes de programación.	13	7	0	0	0
[9]Asequibilidad del Sistema de Tareas para la evaluación de la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación en los primeros años de su carrera.	13	6	2	0	0
[10]Posibilidad de generalización del Sistema de Tareas Docentes Integradoras para la formación de habilidades profesionales en los estudiantes de Ingeniería en Sistemas Computacionales.	13	5	4	0	0

Anexo No. 12
Tabla de Frecuencias Relativas Acumuladas

Aspectos a Valorar en relación con el Sistema de Tareas Docentes Integradoras	ESCALA VALORATIVA				
		BA	A	PA	I
[1]Sistematicidad de las cuatro invariantes funcionales para la formación de la habilidad programar: Concebir el problema, Diseñar el Algoritmo Solución; Traducir lo a un lenguaje de programación y Depurarlo.		0,230769231915474	0,0769230797886849	0	0
[2]Pertinencia de las etapas didácticas declaradas en el Sistema de Tareas Docentes Integradoras		0,307692319154739	0,15384615957737	0	0
[3]Aplicabilidad del Sistema de Tareas Docentes Integradoras en el contexto de las Carreras de Ingeniería en Sistemas Computacionales de las universidades ecuatorianas		0,307692319154739	0,0769230797886849	0	0
[4]El Sistema de Tareas Docentes Integradoras propicia la participación y el protagonismo de los estudiantes en el proceso.		0,461538463830948	0,230769231915474	0	0
[5]Pertinencia de los acciones y operaciones que se proponen en las tareas.		0,384615391492844	0	0	0
[6]Contribución a la formación de la habilidad *programar* desde la integración sistémica e interrelacionada de los contenidos esenciales de las asignaturas de programación en los primeros años de la carrera.		0,230769231915474	0,15384615957737	0	0
[7]El Sistema de Tareas Docentes Integradoras propicia en los estudiantes la transformación del desempeño en la habilidad *programar* a partir de los modos de actuación del		0,230769231915474	0,15384615957737	0	0

Ingeniero de Sistemas.					
[8]Flexibilidad del Sistema de Tareas Docentes Integradoras para la dosificación y tratamiento individual de los niveles de asimilación de contenidos de los estudiantes de programación.		0,538461565971375	0	0	0
[9]Asequibilidad del Sistema de Tareas para la evaluación de la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación en los primeros años de su carrera.		0,461538463830948	0,15384615957737	0	0
[10]Posibilidad de generalización del Sistema de Tareas Docentes Integradoras para la formación de habilidades profesionales en los estudiantes de Ingeniería en Sistemas Computacionales.		0,384615391492844	0,307692319154739	0	0

Anexo No. 13
Tabla de Distribución Inversa Normal Estandarizada

Aspectos a Valorar en relación con el Sistema de Tareas Docentes Integradoras		ESCALA VALORATIVA			
		BA	**A**	**PA**	**I**
[1]Sistematicidad de las cuatro invariantes funcionales para la formación de la habilidad programar: Concebir el problema, Diseñar el Algoritmo Solución; Traducir lo a un lenguaje de programación y Depurarlo.		-0,73631591305856	-1,4260768537263	-3.09	-3.09
[2]Pertinencia de las etapas didácticas declaradas en el Sistema de Tareas Docentes Integradoras		-0,50240219126722	-1,0200762097020	-3.09	-3.09
[3]Aplicabilidad del Sistema de Tareas Docentes Integradoras en el contexto de las Carreras de Ingeniería en Sistemas Computacionales de las universidades ecuatorianas		-0,50240219126722	-1,4260768537263	-3.09	-3.09
[4]El Sistema de Tareas Docentes Integradoras propicia la participación y el protagonismo de los estudiantes en el proceso.		-0,09655860956654	-0,7363159130585	-3.09	-3.09
[5]Pertinencia de los acciones y operaciones que se proponen en las tareas.		-0,29338121379867	-3.09	-3.09	-3.09
[6]Contribución a la formación de la habilidad *programar* desde la integración sistémica e interrelacionada de los contenidos esenciales de las asignaturas de programación en los primeros años de la carrera.		-0,73631591305856	-1,0200762097020	-3.09	-3.09
[7]El Sistema de Tareas Docentes Integradoras propicia en los estudiantes		-0,73631591305856	-1,0200762097020	-3.09	-3.09

la transformación del desempeño en la habilidad *programar* a partir de los modos de actuación del Ingeniero de Sistemas.					
[8]Flexibilidad del Sistema de Tareas Docentes Integradoras para la dosificación y tratamiento individual de los niveles de asimilación de contenidos de los estudiantes de programación.		-0,09655868461895	-3.09	-3.09	-3.09
[9]Asequibilidad del Sistema de Tareas para la evaluación de la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación en los primeros años de su carrera.		-0,09655860956654	-1,0200762097020	-3.09	-3.09
[10]Posibilidad de generalización del Sistema de Tareas Docentes Integradoras para la formación de habilidades profesionales en los estudiantes de Ingeniería en Sistemas Computacionales.		-0,29338121379867	-0,5024021912672	-3.09	-3.09

Anexo No. 14
Encuesta Dirigida a Estudiantes

Objetivo: Caracterizar la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación en los primeros años de la Carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil.

PRESENTACIÓN:

Estimado estudiante:

Con el objetivo de caracterizar la formación de la habilidad *programar* en los estudiantes de Ingeniería en Sistemas Computacionales y su pertinencia en las condiciones de un proceso educativo de calidad en la Universidad de Guayaquil, le pedimos responda con la mayor sinceridad y profundidad el cuestionario que le presentamos.

I.-**DATOS PERSONALES**:

l. l. Género: M _____ F _____

1.2 Edad: _____ años

1.3. Marque con una X si es bachiller con especialización en:

FÍSICO MATEMÁTICO:_____	QUÍMICO BIÓLOGO:____	FILOSÓFICO SOCIAL:___
INFORMÁTICA:____	CONTABILIDAD/COMERCIO:_____	TÉCNICO:____

1.4. Ciclo y año que ingresó a la carrera: ciclo:____ año:____

II. NIVEL DE PROGRAMACIÓN DE COMPUTADORAS

2.1. En qué nivel se cataloga como Programador?

BÁSICO:_____	INTERMEDIO:_____	AVANZADO:_____

2.2. Cuántos lenguajes de programación maneja en la actualidad?

NINGUNO:_____	UNO:____	DOS:_____	MÁS DE DOS:_____

2.3. Con una X, indique cuál es la última asignatura de Programación APROBADA del Pénsum de Ingeniería en Sistemas Computacionales?

PROGRAMACIÓN I ( )	PROGRAMACIÓN II ( )	PROGRAMACIÓN III ( )
PROGRAMACIÓN ORIENTADA A OBJETOS: ( )		

2.4. Indique dentro del paréntesis **el número de veces tomadas** de cada asignatura:

PROGRAMACIÓN I ( )	PROGRAMACIÓN II ( )	PROGRAMACIÓN III ( )
PROGRAMACIÓN ORIENTADA A OBJETOS: ( )		

2.5. Tuvo dificultades en las asignaturas de programación: SI____ NO____

2.6. Si su respuesta anterior fue SI, **ENGLOBE 3 tipos de dificultades fundamentales** que tuvo en las asignaturas de programación:

a) No se le entendía al profesor b) El profesor no sabía la materia

c) No hubo texto guía d) La materia era muy abstracta

e) No sentía motivación por aprender a programar f) No tenía lógica para programar

g) Las clases eran más teóricas que prácticas h) No existieron o hubo poca práctica

i) Aprender un lenguaje de programación fue la prioridad del profesor

j) En el colegio no me enseñaron a programar. k) No aprendí a programar

l) No conoci una metodología sencilla para resolver problemas mediante programación

m) Otros: Especifique:_____

3. PROPUESTA DEL SISTEMA DE TAREAS DOCENTES INTEGRADORAS

MA: Muy de Acuerdo **BA:** Bastante de Acuerdo **A:** Acuerdo **PA**: Parcial Acuerdo **D:** Desacuerdo

ASPECTO A EVALUAR	ESCALA DE VALORACIÓN				
	MA	BA	A	PA	D
3.1. ¿Considera que los contenidos de las asignaturas de programación que usted necesita para aprender a programar están presentes en el Sistema de Tareas que se propone?					
3.2. ¿El Sistema de Tareas Docentes está elaborado con tareas de carácter integrador que interrelacionan los contenidos más esenciales de las asignaturas de programación de computadoras de los primeros años de la carrera de Ingeniería en Sistemas Computacionales?					
3.3. ¿La sistematización de tareas en armonía al resto de las actividades del aula permitió una formación más integral de la habilidad programar?					
3.4. ¿Las tareas integradoras del sistema están estructuradas a partir del nivel de asimilación de los estudiantes desde los contenidos más sencillos a lo más complejos?					
3.5. ¿La aplicabilidad del Sistema de Tareas Docentes integradoras propicia escenarios de práctica y consolidación de conocimientos de los estudiantes?					
3.6. ¿La asequibilidad de la etapa didáctica de Orientación y Familiarización de cada tarea integradora, posibilitó el incremento de motivación e interés por aprender a programar?					
3.7. ¿El Sistema de Tareas Docentes Integradoras prioriza más que la sintaxis de un lenguaje de programación, el aprendizaje de los fundamentos de la lógica de la programación?					
3.8. ¿La factibilidad del sistema de tareas propuesto en la clase, contribuyó a mejorar sus calificaciones y el aprovechamiento académico en las asignaturas de programación de computadoras?					

4. Seleccione por orden de prioridad (escala del 1 al 5) qué elementos requiere el estudiante de Ingeniería de Sistemas en la formación de la habilidad *programar*: **El 1 es el número de mayor valor en su prioridad**.

Aspecto	5	4	3	2	1
4.1. Asimilar los conceptos Básicos de los Paradigmas de Programación ESTRUCTURADA y ORIENTADA A OBJETOS.					
4.2. Dominio de un lenguaje de programación de alto nivel					
4.3. Manejo de herramientas de diseño de algoritmos					
4.4. Manejo de herramientas de modelaje de clases y objetos (UML)					
4.5. Razonamiento Abstracto y Lógica de programación.					
4.6. Análisis del problema a programar					
4.7. Aplicar los Fundamentos de Algorítmica					
4.8. Manejo de lenguaje en pseudocódigo					
4.9.Dominio de la secuencia de pasos para resolver un problema					
4.10Sistematización e Interrelación de los contenidos esenciales de las asignaturas de programación 1,2,3 y O.O.					
4.11.Desarrollo e implementación de proyectos interdisciplinarios de desarrollo de software (matemática, física, química, biología, ingeniería, economía, etc).					
4.12.Pruebas y Depuración de programas					
4.13.Documentación de programas					

Anexo No. 15
Encuesta Dirigida A Profesores

Objetivo: Caracterizar la formación de la habilidad *programar* en los estudiantes de las asignaturas de programación en los primeros años de la Carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil.

Estimado colega:

Con el objetivo de caracterizar la formación de la habilidad *programar* en los estudiantes de Ingeniería en Sistemas Computacionales y su pertinencia en las condiciones de un proceso educativo de calidad en la Universidad de Guayaquil, le pedimos responda con la mayor sinceridad y profundidad el cuestionario que le presentamos.

I.-**DATOS PERSONALES**:

l. l. Género: M _____ F _____

1.2 Tipo de relación docente con la Universidad:

CONTRATO:_____ NOMBRAMIENTO:_____

1.3. Años de servicio de docencia superior en general:_____ años

1.4. Años de docencia en la cátedra de PROGRAMACIÓN: _____ años

II,. ASPECTOS DE ENSEÑANZA DE LA PROGRAMACIÓN DE COMPUTADORAS

ASPECTOS	ESCALA VALORATIVA		
	SIEMPRE	ALGUNAS VECES	NUNCA
2.1. ¿El Sistema propuesto propicia el desarrollo de tareas integradoras desde la familiarización, reproducción hasta la producción y creación?			
2.2. ¿Con el Sistema de Tareas Docentes Integradoras se percibe en los estudiantes rasgos de una mejor motivación e interés por programar?			
2.3. ¿En la propuesta se estructuran tareas integradoras conducentes a potenciar más la lógica de la programación que la enseñanza de reglas sintácticas de un lenguaje de programación?			
2.4. ¿Las tareas integradoras demuestran una síntesis de los contenidos esenciales de las asignaturas de programación de computadoras, necesarios para enseñar a programar con cierto nivel de generalización?			
2.5. ¿El Sistema de Tareas Docentes Integradoras propician espacios y actividades de práctica y laboratorios de programación?			
2.6. ¿Las 4 tareas integradoras que presenta el sistema propuesto posibilitan un vínculo holístico entre lo instructivo y educativo?			

3. PROPUESTA DEL SISTEMA DE TAREAS DOCENTES INTEGRADORAS

MA: Muy de Acuerdo **BA:** Bastante de Acuerdo **A:** Acuerdo **PA**: Parcial Acuerdo **D:** Desacuerdo

ASPECTO A EVALUAR	ESCALA DE VALORACIÓN				
	MA	BA	A	PA	D
3.1. Si considera que los contenidos que usted necesita para programar están presentes en el sistema de tareas que se propone?					
3.2. En qué medida se aprendió a programar más y mejor con el Sistema de Tareas propuesto?					
3.3. Considera Ud. que el Sistema de Tareas está elaborado con lo esencial de los contenidos de las asignaturas de programación?					
3.4. Las acciones y operaciones planteadas en el Sistema de Tareas, les permitió recorrer por conocimientos de los más fáciles a los más complejos?					

4. Seleccione por orden de prioridad (escala del 1 al 5) qué elementos requiere el estudiante de Ingeniería de Sistemas en la formación de la habilidad *programar*: **El 1 es el número de mayor valor en su prioridad**.

Aspecto	5	4	3	2	1
4.1.Asimilar los conceptos Básicos de los Paradigmas de Programación ESTRUCTURADA y ORIENTADA A OBJETOS.					
4.2.Dominio de un lenguaje de programación de alto nivel					
4.3.Manejo de herramientas de diseño de algoritmos					
4.4. Manejo de herramientas de modelaje de clases y objetos (UML)					
4.5.Razonamiento Abstracto y Lógica de programación.					
4.6.Análisis del problema a programar					
4.7.Aplicar los Fundamentos de Algorítmica					
4.8.Manejo de lenguaje en pseudocódigo					
4.9.Dominio de la secuencia de pasos para resolver un problema					
4.10. Sistematización e Interrelación de los contenidos esenciales de las asignaturas de programación 1,2,3 y O.O.					
4.11.Desarrollo e implementación de proyectos interdisciplinarios de desarrollo de software (matemática, física, química, biología, ingeniería, economía, etc).					
4.12.Pruebas y Depuración de programas					
4.13. Documentación de programas					

Anexo No. 16
Guía para la Observación a Clases

A continuación se presenta la información que se ha considerado necesaria para realizar mediante la Observación a Clases, una valoración de la aplicación del Sistema de Tareas Docentes Integradoras propuesto para la formación de la habilidad *programar* en los estudiantes de los primeros años del Periodo Lectivo 2013 – 2014 en la carrera de Ingeniería en Sistemas Computacionales de la Universidad de Guayaquil.

Objetivos de la Guía:

Valorar la aplicación del Sistema de Tareas Docentes Integradoras mediante la observación de clase a los profesores de las asignaturas de programación de computadoras en los primeros años de carrera.

Los **Aspectos** considerados en la Observación de Clase en las etapas didácticas, son:

- **Pertinencia** del Sistema de Tareas con la formación de la habilidad *programar*
- **Factibilidad** de las etapas del Sistema de Tareas
- **Aplicabilidad** del Sistema de Acciones y Operaciones y su **ejecución** en vista a la formación de la habilidad *programar.*
- **Asequibilidad** del Sistema de Tareas
- **Sistematicidad** en clase de las tareas para formar la habilidad *programar.*
- **Valor instructivo y educativo** del Sistema de Tareas en la formación de los estudiantes.

La **Escala de Valoración** utilizada consta de 5 niveles: *Muy Adecuado* MA, *Bastante Adecuado* BA, *Adecuado* A, *Poco Adecuado* PA, *Inadecuado* I.

Muy Adecuado MA: Se considera aquel aspecto que es Optimo, en el cual se expresan todas y cada una de las propiedades, consideradas como componentes esenciales para determinar la calidad del Sistema de Tareas Docentes Integradoras propuesto.

Bastante Adecuado BA: Se considera aquel aspecto que expresa en casi toda su generalidad las cualidades esenciales del Sistema de Tareas Docentes Integradoras propuesto siendo capaz de representar con un grado bastante elevado, los rasgos fundamentales que tipifican su calidad.

Adecuado A: Se considera aquel aspecto que tiene en cuenta una parte importante de las cualidades del Sistema de Tareas Docentes Integradoras propuesto, las cuales expresan elementos de valor con determinado nivel de suficiencia.

Poco Adecuado PA: Se considera aquel aspecto que expresa un bajo nivel de adecuación en relación con el estado deseado del Sistema de Tareas Docentes Integradoras propuesto, al expresarse carencias en determinados componentes, considerados esenciales para determinar su calidad.

Inadecuado I: Se considera aquel aspecto en el que se expresan marcadas limitaciones y contradicciones que no le permiten adecuarse a las cualidades esenciales que determinan la calidad del Sistema de Tareas Docentes Integradoras propuesto, por lo que no resulta procedente su aplicación.