Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

BUENAS PRÁCTICAS DENTRO DEL MARCO DE UNA METODOLOGÍA DE DESARROLLO DE SOFTWARE PARA EL USO DE LOS ESTUDIANTES DE INGENIERÍA EN CIENCIAS Y SISTEMAS DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Teresa Domicila Quezada Mauricio

Asesorado por el Ing. José Ricardo Morales Prado

Guatemala, agosto de 2014

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

BUENAS PRÁCTICAS DENTRO DEL MARCO DE UNA METODOLOGÍA DE DESARROLLO DE SOFTWARE PARA EL USO DE LOS ESTUDIANTES DE INGENIERÍA EN CIENCIAS Y SISTEMAS DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

TERESA DOMICILA QUEZADA MAURICIO
ASESORADO POR EL ING. JOSÉ RICARDO MORALES PRADO

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, AGOSTO DE 2014

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Narda Lucía Pacay Barrientos
VOCAL V	Br. Walter Rafael Véliz Muñoz
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. César Augusto Fernández Cáceres
EXAMINADOR	Ing. Juan Álvaro Díaz Ardavín
EXAMINADOR	Ing. José Alfredo González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

BUENAS PRÁCTICAS DENTRO DEL MARCO DE UNA METODOLOGÍA DE DESARROLLO DE SOFTWARE PARA EL USO DE LOS ESTUDIANTES DE INGENIERÍA EN CIENCIAS Y SISTEMAS DE LA UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 10 de marzo de 2013.

Teresa Domicila Quezada Mauricio

ACTO QUE DEDICO A:

Dios	Por poner en mi camino a todas esas personas que me guiaron y contribuyeron para que pudiera alcanzar este logro.
Mi madre	Porque a ella debo todo lo que soy, lo que sé y lo que tengo, ella ha sabido mostrarme el camino y me ha dado las herramientas para conquistarlo, supo darme valores y nunca me ha dejado claudicar de mis sueños y metas. Gracias a sus enseñanzas y consejos hoy puedo presentar este trabajo de investigación.
Mi abuela	Quién me apoyó, cuidó y consintió durante todos estos años, me ayudó a forjar mi carácter y principios.
Mis hermanos	Ángel y Mynor Quezada Mauricio, quienes han sido fuente de inspiración para no claudicar en este proceso de formación académica.
Mis amigos	En quienes he encontrado la ayuda, el apoyo y cariño siempre.

AGRADECIMIENTOS A:

La Universidad de San Carlos de Guatemala	Por brindarme la oportunidad de formar parte de esta gloriosa Universidad.
Facultad de Ingeniería	Por ser la casa de estudios que me acogió durante todos estos años, brindándome el ambiente y los recursos para adquirir los conocimientos en el área de Ingeniería en Ciencias y Sistemas que hoy poseo.
Mis amigos de la Facultad	Adonai Navas, Allan Solís, Alejandro Monterroso, Gianni Leiva, Julio Gonzáles y Mario Gaitán.
Ing. Ricardo Morales	Por su comprensión y colaboración en todo momento para el desarrollo de este trabajo de investigación.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

Administración Ciencia social que estudia las técnicas de administración, asimismo consiste en los procesos de planificación, organización, dirección y control de los recursos de una organización o proyecto.

Arquitectura Se refiere al término arquitectura dentro del contexto de arquitectura de software, es decir, la estructura general de un sistema donde se definen los módulos principales, las funciones de cada uno de estos módulos y el flujo de información entre ellos.

Buenas prácticas Acción o conjunto de acciones que, fruto de la identificación de una necesidad, son sistemáticas, eficaces, eficientes, sostenibles, flexibles, y están pensadas y realizadas por los miembros de una organización con el apoyo de sus órganos de dirección, y que, además de satisfacer las necesidades y expectativas de sus clientes, suponen una mejora evidente de los estándares del servicio, siempre de acuerdo con los criterios éticos y técnicos de la empresa y alineadas con su misión, su visión y sus valores.

Control de calidad	Proceso en el cuál se busca garantizar la calidad del software, es decir, consiste en una serie de revisiones y pruebas llevadas a cabo durante el ciclo de vida del software, para asegurar que cada uno de los productos y subproductos cumplan con las características de calidad establecidas.
Metodología de desarrollo	Conjunto ordenado de pasos a seguir para llegar a la solución de un problema u obtención de un producto o software; metodología es el conjunto de métodos que se utilizan en la realización de un software en general.
Patrones de diseño	Serie de soluciones a problemas que se presentan de manera recurrente, las cuales están basadas en experiencias anteriores. Según Allen Holub, son una técnica para resolver problemas de clases.
Proceso	Conjunto de fases o etapas ordenadas que se siguen para la realización de una actividad. En este caso, generalmente se utilizará en el contexto del proceso de desarrollo de software y proceso de diseño de software, que no es más que el conjunto de fases o etapas a seguir para la realización del desarrollo o del diseño de un software.

RESUMEN

Durante el siguiente trabajo de investigación, se analiza la innovación y mejora del proceso de desarrollo de software que se aplica en la carrera de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala por los estudiantes de la misma. Para esto se incorporan criterios de análisis acerca de la deficiencia en la implementación de una metodología de desarrollo de software, ampliando su alcance a las buenas prácticas de desarrollo de software necesarias; para la buena aplicación de la misma; asegurando así un producto final de una mejor calidad. La falta de aplicación consistente de algunas buenas prácticas dentro del proceso de desarrollo y ejecución de proyectos de software en la carrera, se refleja en la calidad de los productos finales de los proyectos que se desarrollan en la carrera.

Para poder determinar si existe la necesidad de aplicar correctamente una metodología de desarrollo y las buenas prácticas básicas dentro del proceso de desarrollo de software, se realizó una encuesta a los estudiantes de los cursos del área de desarrollo de software, para conocer su opinión y conocer su perspectiva respecto a las metodologías de desarrollo utilizadas hasta el momento, además de sostener entrevistas con catedráticos de algunos cursos y con el director de la Escuela de Ingeniería en Ciencias y Sistemas. A partir de los resultados obtenidos, se han podido determinar las debilidades y las fortalezas del proceso de desarrollo y ejecución de proyectos de software en la carrera.

Luego del análisis del contenido de los cursos del área de desarrollo de software, se realizarán una serie de encuestas a estudiantes de los cursos

evaluados, en busca de las prácticas que implementan en el desarrollo de software, buenas o malas prácticas para identificar necesidades y la situación actual de los cursos en relación a las buenas prácticas necesarias para el desarrollo de software de calidad. La necesidad de generar nuevos paradigmas en la ingeniería de software, requiere el desarrollo de modelos y metodologías que sean utilizadas adecuadamente. Se propone una metodología tomando como base los modelos de la ingeniería del software, los elementos propios del desarrollo de proyectos de software dados en la Facultad de Ingeniería.

La siguiente etapa de la investigación consiste en el estudio de estrategias para el desarrollo de software, esto mediante la comparación de las prácticas existentes y la comparación de los procesos actuales con estándares de desarrollo de software. Las estrategias que se buscan durante el presente trabajo de investigación, consisten en el proceso para la aplicación de buenas prácticas y el adecuado desarrollo de software mediante la utilización de metodologías de desarrollo, dando un especial énfasis en el análisis de los requerimientos de los proyectos y la planificación del desarrollo, así como el desarrollo de pruebas. Además de la implementación de técnicas de diseño acorde al contenido del curso y los objetivos de los proyectos a desarrollar.

La última etapa consiste en la elaboración de recomendaciones para que puedan ser utilizadas por los catedráticos y por personal de la Escuela de Ingeniería en Ciencias y Sistemas, como herramienta de apoyo a la reforma curricular del área de desarrollo de software.

OBJETIVOS

General

Proporcionar una herramienta de apoyo para los estudiantes con las estrategias de implementación de una metodología de desarrollo de software de manera integral, poniendo especial cuidado en las buenas prácticas en el proceso de desarrollo de los proyectos de software de la carrera de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala.

Específicos

1. Determinar las buenas prácticas de desarrollo de software, actualmente impartidas en los cursos del área de desarrollo de software con base en estándares.

2. Determinar las necesidades dentro de los cursos del área, buenas prácticas para mejorar la calidad del desarrollo de software en los cursos del área de desarrollo de software.

3. Elaborar una propuesta de implementación de buenas prácticas dentro del contenido de los cursos del área de desarrollo de sistemas.

Hipótesis

La falta de buenas prácticas en el proceso de desarrollo de software utilizado por los estudiantes de Ingeniería en Ciencias y Sistemas, y la mala aplicación de una metodología para el desarrollo de software, se convierten en una deficiencia en el aseguramiento de la calidad del producto final como tal y del proceso de gestión del proyecto. Por tal razón, se hace necesario establecer un proceso para la correcta implementación de una metodología de desarrollo de software ampliado, con una serie de buenas prácticas adecuadas al proceso de desarrollo de software en proyectos del área de desarrollo de software de la carrera; de esta manera obtener un desarrollo más limpio, confiable y de mayor calidad.

Hipótesis nula:

La aplicación de metodologías de desarrollo de software y buenas prácticas de desarrollo, no constituye una deficiencia para garantizar la calidad de los proyectos del área de desarrollo de software de la carrera de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala.

INTRODUCCIÓN

Durante el siguiente modelo de investigación, se incorporan criterios de análisis crítico acerca de la carencia de una metodología para el análisis y diseño de proyectos de software y además la falta de implementación de buenas prácticas en el desarrollo de software, en la Escuela de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala.

En la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, se realiza una serie de proyectos que representan la práctica y aplicación de los conocimientos adquiridos en los cursos de la carrera, muchos de ellos son proyectos de construcción de software desarrollados en períodos cortos y medianos de tiempo.

Durante el proceso de desarrollo, se hace necesaria la implementación de buenas prácticas de desarrollo de sistemas para poder llegar a cumplir de manera satisfactoria con los objetivos del curso y también con los requerimientos del proyecto en sí.

Para el desarrollo de este trabajo de investigación se realizará un diagnóstico de prácticas impartidas e implementadas durante los diferentes años de la carrera, en el área de desarrollo de software y proponer un sencillo proceso de implementación de buenas prácticas, que contribuya a la mejora e innovación del proceso de desarrollo de software impartido en la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala.

.

1. MARCO METODOLÓGICO

El presente estudio se sitúa en una perspectiva metodológica de investigación interpretativa, utilizando la teoría de decisión basada en el comportamiento, donde se analizará el comportamiento de las diferentes variables y los resultados de las mismas, para establecer un proceso de enseñanza y aprendizaje del desarrollo de software adecuado para la carrera de Ingeniería en Ciencias y Sistemas, haciendo énfasis en la aplicación de buenas prácticas en el desarrollo de proyectos de software.

La metodología para la elaboración del presente trabajo de investigación, consiste en el desarrollo por etapas, con resultados incrementales para la investigación. El mismo se sitúa en una perspectiva metodológica de investigación interpretativa, utilizando la teoría de decisión basada en el comportamiento, donde se analizará el comportamiento de las diferentes variables y los resultados de las mismas, para establecer un proceso de desarrollo de software adecuado como apoyo a los estudiantes de la carrera de Ingeniería en Ciencias y Sistemas, haciendo énfasis en la aplicación de buenas prácticas que garanticen la calidad en el desarrollo de proyectos de software y el producto final.

El diseño de la investigación es de tipo estudio de casos, a partir del cual el enfoque de recolección y análisis de datos se ha definido como mixto, de carácter cualitativo y cuantitativo, en el que se integran diversas técnicas.

Metodológicamente, comprende 3 grandes etapas, por un lado el análisis del contenido de los cursos del área de ingeniería de software; de esta manera

poder situar esta investigación y poder visualizar el objetivo de la misma de una manera más clara. La segunda etapa, consiste en la investigación en relación con las metodologías implementadas por los estudiantes dentro de la Facultad de Ingeniería, de esta manera poder analizar y darle a los estudiantes una mejor perspectiva al respecto, además de las recomendaciones de catedráticos e ingenieros egresados de la carrera. Por último, la tercera etapa consiste en la elaboración de una propuesta para la implementación de las buenas prácticas dentro del área de desarrollo de software con la aplicación de una metodología, proponiendo para ello el proceso de aplicación de 2 metodologías de desarrollo una ágil y otra rígida.

Dichas etapas son descritas a continuación:

1.1. Análisis del contenido programático de cursos

Esta etapa consiste en recopilar y estudiar la información principal de los cursos del área de desarrollo, los cuales serán analizados mediante una ficha de resumen con base en el programa general del curso.

Tabla I. **Resumen de cursos**

Código de curso	Código de curso
Nombre:	Nombre del curso
Objetivos:	Objetivos del curso
Descripción:	Temas del curso
Metodología:	Metodología del curso
Contenido:	Temas principales del curso
Bibliografía:	Referencias Bibliográficas

Fuente: elaboración propia.

1.2. Entrevista con catedráticos e ingenieros en Ciencias y Sistemas egresados de la Universidad de San Carlos de Guatemala

Se llevarán a cabo entrevistas con algunos catedráticos del área de desarrollo de sistemas, para conocer sus opiniones referentes al proceso de desarrollo y escuchar sus recomendaciones y comentarios. También se llevará a cabo entrevistas a ingenieros en Ciencias y Sistemas egresados de la Universidad de San Carlos de Guatemala con experiencia en el desarrollo de proyectos de software en sus empleos.

De esta manera poder observar el panorama desde ámbitos reales de proyectos de desarrollo de software, con mayores exigencias y responsabilidades. Desde estos 2 puntos de vista, conocer los problemas comunes en el proceso de desarrollo y recoger experiencias de los tipos de soluciones a diferentes problemas durante el desarrollo.

Para esta etapa las entrevistas han sido personales y de forma directa, con un esquema abierto alrededor de tres preguntas básicas.

Tabla II. **Ficha básica de entrevista a catedráticos**

	Pregunta
Entrevista	¿Cuál es su opinión acerca del proceso de desarrollo que se aprende en la carrera de Ingeniería en Ciencias y Sistemas?
	¿Con base a su experiencia qué recomendaciones daría para mejorar el proceso de desarrollo actual que siguen los alumnos de la Facultad de Ingeniería?
	¿Qué problemas ha enfrentado y cómo dio solución a los mismos en el proceso de desarrollo de proyectos de software en su experiencia laboral?

Fuente: elaboración propia.

Con base en los datos recabados durante la entrevista, se realiza una tabla con las recomendaciones generales *versus* los problemas a los que se han enfrentado los entrevistados.

1.3. Encuesta a los estudiantes

Para poder conocer la visión de los estudiantes, sus opiniones y realidades en el proceso de desarrollo y las prácticas que utilizan, para ello se realiza una encuesta en la que se reflejan los hábitos de desarrollo de software.

Con la información que se obtenga de esta etapa, será posible diseñar y sugerir una estrategia de implementación de buenas prácticas en el desarrollo de software, además de la utilización de una metodología de desarrollo. Para la realización de esta encuesta se diseñó una, la cual se detalla en la tabla VI.

1.4. Encuesta a ingenieros recién graduados de la Facultad de Ingeniería

Para conocer el punto de vista respecto al proceso de desarrollo y a la necesidad de metodologías de desarrollo de la carrera de la Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala, se utilizó los resultados de la encuesta 53963, sobre la evaluación del *pensum* de estudios de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala, respecto del perfil del egresado; dirigida a ingenieros recién graduados en los últimos 5 años, y cuyos resultados reflejan tanto la opinión acerca del proceso de desarrollo en la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, como la experiencia en el campo laboral en relación al mismo tema. Para detalles ver la tabla VII.

2. MARCO TEÓRICO

2.1. Teoría de la decisión basada en el comportamiento Behavioral Decisión Theory (BDT)

Para el desarrollo del presente trabajo de investigación se ha basado en la teoría de decisión del comportamiento, la cual por sus siglas en inglés es conocida como BDT. Esta es una teoría descriptiva que se basa en la toma de decisiones humanas, esto indica que luego del estudio preliminar se tendrá una serie de opciones, de las cuales se debe tomar una desición de acuerdo a las necesidades que se detecten.

Esta teoría está basada en la teoria de la decisión, se inicia con una teoría normativa tradicional de toma de decisiones racional, la toma por ejemplo; de decisiones con base en reglas bayesianas.

Es decir esta teoria concierne a la identificación de la mejor decisión que pueda ser tomada, con lo que de acuerdo a un consenso entre los catedraticos de los cursos, se tomará la decisión y definirá las necesidades, posteriormente las estrategias y finalmente elaborar las recomendaciones al curso; asumiendo que una persona que tenga que tomar decisiones (decisión *maker*) sea capaz de estar en un entorno que completa la información, capaz de calcular con precisión y completamente racional.

La aplicación práctica de esta aproximación prescriptiva (de como la gente debería hacer y tomar decisiones) se denomina análisis de la decisión y proporciona una búsqueda de herramientas, metodologías y software para

ayudar a las personas a tomar mejores decisiones. Las herramientas de software orientadas a este tipo de ayudas, se desarrollan bajo la denominación global de sistemas para la ayuda a la decisión (*decision support systems)*, abreviado en inglés como DSS.

La idea es comprender e incorporar patrones de toma de decisión real de los seres humanos durante el desarrollo de proyectos de software, por ejemplo; la toma de desiciones, la infraponderación o sobreponderación de las probabilidades, la decisión o elección de encuadre, la elección de satisfacer, para describir la decisión real sobre tendencias. Estas tendencias, una vez entendida, entonces se pueden abordar, por ejemplo, mediante el diseño e intervenir con sistemas apropiados de apoyo a las decisiones.

Existen tipos de decisión vistos desde el punto de vista del desarrollo de una teoría, estos son:

- Decisión sin riesgo entre insumos y unidades, las cuales son inconmensurables (unidades que no pueden ser medidas bajo las mismas unidades)
- Elección bajo impredecibilidad
- Elección intertemporal - estudio del valor relativo que la gente asigna a dos o más unidades en diferentes momentos del tiempo
- Decisiones sociales: decisiones tomadas en grupo o bajo una estructura organizativa

2.1.1. Proceso de decisión

Una subdivisión del proceso de decisión, fue propuesta por Brim et al. (1962, p. 9). Dividieron el proceso de decisión en los siguientes 5 pasos:

- Identificación del problema
- La obtención de la información necesaria
- Producción de soluciones posibles
- Evaluación de tales soluciones
- La selección de una estrategia de actuación y rendimiento.

2.2. Proceso de desarrollo de software

En 1968 NATO definió la ingeniería del software como el establecimiento y uso de principios de ingeniería razonables, con el objetivo de obtener software realizado de manera óptima, es decir eficiente y eficaz; lo cual significa que debe tener características como utilizar la menor cantidad de recursos para su realización y que cumpla con todos los requerimientos esperados por el cliente. En 1985, Fairley describió la ingeniería de software como la disciplina tecnológica concerniente a la producción y mantenimiento sistemático de productos de software, que son desarrollados y modificados en el tiempo y con los costes estimados. Además, la ingeniería del software tiene que ver con cuestiones de gestión, que caen fuera del dominio de la programación tradicional.

Como se puede observar, la ingeniería de software es un proceso en el cual se desarrolla software, se define el qué, quién, cuándo y cómo del desarrollo de software. Específicamente para el proceso de desarrollo de software, es necesario considerar las 4 actividades básicas del mismo:

- Especificación del software
- Desarrollo del software
- Validación del software
- Evolución del software

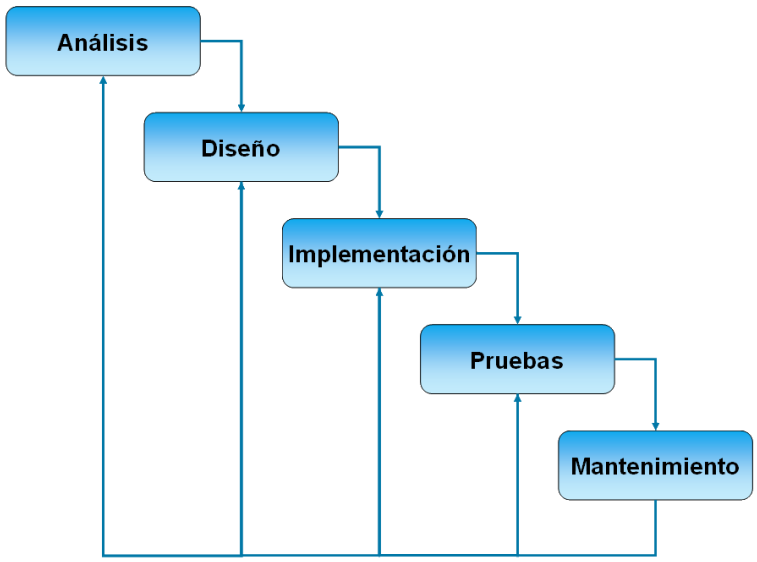
Por otro lado es importante saber que para cualquier ingeniería y muchas otras áreas, es recomendable considerar las 3 técnicas para el tratamiento de proyectos:

- El modelado, se refiere a la abstracción y simplificación del objeto o solución en el mundo real, pero que es suficientemente realista como para dar una idea de lo que ocurrirá en la realidad y usarse como base del desarrollo.
- La división del producto, consiste en el fraccionamiento del producto, de modo que este pueda asignarse como tarea a un miembro del equipo.
- La división del proceso, significa que se planifica el desarrollo del producto por etapas o fases, lo cual hace un proceso más organizado.

Cuando se ha modelado el problema y dividido un proyecto en miniproyectos, más fáciles de manejar y completar. Se debe considerar cada miniproyecto como una iteración y cada iteración contiene todos los elementos de un proyecto común.

- Planificación
- Análisis y diseño
- Construcción
- Integración y pruebas
- Versión del producto (interna o externa)

Figura 1. **Ciclo de vida del software Modelo en Cascada**



Fuente: http://es.wikipedia.org/wiki/Desarrollo_en_cascada.Consulta: 1 de mayo de 2013 .

Estos elementos del proyecto pueden variar y ser discutidos por los expertos del tema, orden de aplicación, los requisitos previos a cada uno de ellos como la forma en que estos son aplicados. Todo esto está contenido dentro del ciclo de vida del software para el cual existen diferentes modelos. Por ejemplo de Tipo V, Espiral y Cascada.

2.3. Normas internacionales aplicables al proceso de desarrollo de software

En las diferentes etapas del ciclo de vida del software, puede aplicarse una serie de normas internacionales, que describen el estándar que cumple el proceso y el producto final; algunas de estas normas se especifican en la siguiente tabla:

Tabla III. **Normas internacionales en función de la etapa del proceso de desarrollo**

REQUERIMIENTOS	• IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications • IEEE Std 1063-2001 IEEE Standard for Software User Documentation
ANÁLISIS Y DISEÑO	• IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software Intensive • IEEE Std 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems
CONSTRUCCIÓN DEL SOFTWARE	• ANSI/IEEE 1008-1987 IEEE Standard for Software Unit Testing
PRUEBAS DEL SOFTWARE	• IEEE Std 1012-1998 IEEE Standard for Software Verification and Validation • IEEE Std 730™-2002 IEEE Standard for Software Quality Assurance Plans • IEEE Std 829-1998 IEEE Standard for Software Test Documentation
INTEGRACIÓN DEL SOFTWARE	• ISO 12207
MANTENIMIENTO DEL SOFTWARE	• IEEE Std 219-1998 IEE Standard for Software Maintenance

Fuente: elaboración propia.

2.4. Contexto de la investigación

A lo largo de la carrera de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala, tiene lugar una serie de proyectos de desarrollo de software, en los que se aplican conocimientos adquiridos durante los cursos de la carrera; el ciclo de vida de estos proyectos da lugar a la necesidad de un mejor control y administración de dichos proyectos, asimismo la garantía de resultados satisfactorios y de calidad.

Por tal razón, es necesario considerar la implementación de una metodología de desarrollo de software, durante el proceso de desarrollo de dichos proyectos por parte de los estudiantes de la carrera en el transcurso del área de desarrollo de software, especialmente en aquellos cursos en los que se tenga como tema la implementación de metodologías de desarrollo y buenas prácticas de desarrollo de software.

2.5. Diseño de la muestra

El diseño de muestra corresponde a la tomada para la realización de la encuesta, dirigida a los estudiantes de la carrera, dicho modelo se describe dentro de los instrumentos utilizados para el análisis del presente trabajo de investigación.

2.5.1. Configuración y/o preparación del marco de muestreo

A partir de la información recabada en los diferentes tipos de encuestas realizadas, se tomará el número total de estudiantes en los cursos del área de desarrollo de software para su ulterior muestreo.

2.5.2. Identificación de las variables principales y tipo de muestreo

Dadas las condiciones del presente estudio, se hace necesario hacer una aclaración preliminar antes de identificar plenamente las variables principales a medir.

Se tiene principalmente dentro de este estudio a 2 poblaciones distintas de estudiantes del área de desarrollo de software, a saber: estudiantes no

egresados y estudiantes egresados, luego se harán muestreos paralelos en ambas poblaciones y por tanto nuestras variables principales son las encuestas mismas realizadas a cada población. Por otra parte, en cuanto al tipo de muestreo que se propone por el tipo de datos a obtener, se considerará con las siguientes características:

- Muestreo no probabilístico por conveniencia: ya que se supondrá una distribución normal en ambas poblaciones, seleccionando además los elementos a medir de forma arbitraria, debido al tamaño real de las poblaciones en estudio. La conveniencia en este caso, está determinada por un análisis de la población estudiantil no egresada, y una encuesta ya realizada a la población estudiantil egresada.

- Monoetápico para variables dicotómicas: se toma monoetápico, ya que será una sola etapa en la toma de datos y se consideran como variables dicotómicas, ya que la propiedad que se está buscando en las variables a medir, son de que considere que se realizan buenas prácticas durante la enseñanza en los cursos del área de desarrollo de software o no, por tanto las variables se consideran dicotómicas.

- Marco muestral en forma de listado: tal listado lo conforman todas las encuestas del estudio.

2.5.3. Definición de dominios de estudio y cobertura de la muestra

El diseño específico de cada encuesta contempla a cada población de estudiantes, por lo que el dominio de estudio en las encuestas específicas a cada población estudiantil y la cobertura de la muestra, será en la Escuela de

Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala y egresados de la misma.

2.5.4. Definición de los estimadores

Por la escogencia del tipo de muestreo no probabilístico, la confiabilidad no se puede medir teóricamente, se considera luego innecesario definir estimadores.

2.5.5. Asignación y discusión de parámetros muestrales básicos

Se utilizarán los parámetros propios del muestreo simple aleatorio monoetápico para variables dicotómicas, definido dentro de la misma encuesta, en relación al tipo de información que se solicita en la pregunta.

2.5.6. Algoritmo adecuado para cálculo de tamaño de muestra

En el presente diseño se utilizará la siguiente fórmula que guarda la proporcionalidad para calcular la muestra en los distintos grupos a estudiar.

$$n = \frac{z^2.p.q.N}{N.E^2 + z^2.p.q}$$

Donde:

n = es el tamaño de la muestra resultante

z = es el estadígrafo para un nivel de significación α (alfa) para la distribución normal (en este caso particular se toma el usual 1,96)

p = es la probabilidad de ocurrencia (en este caso se toma como 0,5)

q = es la probabilidad de no ocurrencia (1-p), en este caso es igual a 0,5

E = es el error permitido (en este caso suponemos un error máximo del 5 por ciento)

N = es el tamaño de la población a muestrear

2.5.7. Afijación de la muestra en dominios de estudio

Se tomará proporcional al tamaño de la población total, es decir una proporción simple que se guardará en cada una de las poblaciones de estudio y sus respectivas encuestas.

2.5.8. Método de selección de las unidades primarias

Se tomará como UPM o unidades primarias de muestra a cada una de las encuestas o registros llenados por cada estudiante (según la población específica, anteriormente delimitada).

2.5.9. Tendido de la muestra y plan de control muestral de campo

Ambos, el tendido de la muestra y el plan de control muestral, estarán a cargo de Teresa Domicila Quezada Mauricio, y se estará utilizando herramientas virtuales que contribuyan al tendido y control de la muestra.

2.5.10. Probabilidades de selección y expansores

No se hace necesario la expansión de la muestra, debido a que se ha utilizado un criterio específico de selección, y este es representativo misma en función del tamaño de la población y la muestra definida.

2.5.11. Determinación y selección del software de cálculo

Se pueden realizar los distintos cálculos con Microsoft Excel o cualquier software para el manejo de bases de datos que por el tamaño muestral, no se requiere de un software muy complejo o tecnología especifica.

2.5.12. Establecimiento de los niveles de desagregación de resultados

Las desagregaciones posibles estarán definidas a partir de cada una de las preguntas, y serán cuantitativas o cualitativas, según sea el caso y el objeto de la pregunta.

2.5.13. Plan de ajuste de precisiones esperadas

Dichos ajustes estarán en función de la calidad de información que se pueda generar, teniendo un control específico sobre los posibles faltantes, que al respecto se considerarán fuera de la muestra, es decir, en aras de la precisión de los datos, se tomarán como muestras las encuestas debidamente llenadas.

Las etapas a realizar son descritas a continuación:

2.6. Análisis del contenido programático de cursos

Esta etapa consiste en recopilar y estudiar la información principal de los cursos del Área de Desarrollo, los cuales serán analizados mediante una ficha de resumen con base en el programa general del curso.

Tabla IV. **Ficha resumen de cursos**

Código de curso	Código de curso
Nombre:	Nombre del curso
Objetivos:	Objetivos del curso
Descripción:	Temas del curso
Metodología:	Metodología del curso
Contenido:	Temas principales del curso
Bibliografía:	Referencias Bibliográficas

Fuente: elaboración propia.

2.7. Entrevista con los catedráticos del curso

Se llevaran a cabo entrevistas con los catedráticos de los cursos analizados, para conocer sus opiniones referentes al curso y al área de desarrollo en general, y poder de estas entrevistas empezar a conocer fortalezas y debilidades de los cursos. Para esta etapa se ha elaborado una ficha para recopilar datos.

Tabla V. **Ficha de entrevista**

Código de curso	Código de curso
Nombre curso:	Nombre del curso
Catedrático:	Nombre del catedrático
Fortalezas del contenido:	Opinión de las fortalezas del curso
Debilidades del contenido:	Opinión de las debilidades del curso
Opiniones generales:	Opiniones generales del contenido del curso
Recomendaciones:	Recomendaciones

Fuente: elaboración propia.

De los datos recabados, debe de tabularse y encontrar las debilidades y necesidades del curso, para poder buscar la manera de implementar un plan de mejoras.

2.8. Encuesta a los estudiantes

Para poder conocer la visión de los estudiantes y las prácticas que utilizan, es necesario realizar una encuesta en la que se reflejen los hábitos de desarrollo de software. Con la información que se obtenga de esta etapa, será posible diseñar y proponer una estrategia de implementación de buenas prácticas en el desarrollo de software. Para la realización de esta encuesta se ha diseñado la siguiente ficha técnica.

Tabla VI. **Ficha técnica de la encuesta a estudiantes**

No.	Pregunta
1	En una escala del 1 al 10, dónde 10 es “muy interesante” y 1 es “nada interesante” ¿Qué tan interesante le parece el tema “Las buenas prácticas de desarrollo de software”?
2	¿Cuál o cuáles de las siguientes características le atraen del tema de implementación de buenas prácticas en el desarrollo de software?
3	¿En qué cursos de la carrera le han enseñado acerca de buenas prácticas para el desarrollo de software?
4	¿En qué cursos se debería de reforzar la enseñanza de buenas prácticas para el desarrollo de software?
5	De las siguientes buenas prácticas… ¿Cuáles son las que comúnmente utiliza para el desarrollo de software?
6	¿Cuál o cuáles de las siguientes características reflejan de alguna manera su realidad en cuanto a la implementación de las buenas prácticas de desarrollo de software?
7	¿Cuál o cuáles de las siguientes prácticas le parece importante utilizar para el desarrollo de software?
8	¿Creé que la aplicación integrada de las prácticas, en el contexto de una metodología de desarrollo, debe de ser parte del contenido y prácticas de los cursos de desarrollo de software?
9	¿Cuáles de las siguientes características creé usted que describe una metodología ágil?
10	¿Cuáles de estas características corresponden a una metodología tradicional o rígida?
11	¿Qué metodología(s) de software considera usted que deben de enseñarse y aplicarse en los distintos años de la carrera?
12	¿Tiene algún comentario o sugerencia para mejorar la calidad de software que se realiza durante los diferentes años de la carrera?

Fuente: elaboración propia.

2.9. Encuesta a ingenieros recién graduados de la Facultad de Ingeniería

Para conocer el punto de vista de algunas de las personas recién graduadas de la carrera de la Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala, se utilizó datos de una encuesta realizada por el señor Juan José Armas, a algunos ingenieros graduados en los últimos 5 años.

Tabla VII. **Ficha técnica de la encuesta a egresados**

Año de ingreso a la Facultad de Ingeniería
Año de cierre de cursos
Año de graduación
Sexo
Área de actividades en la actualidad
Dónde trabaja
Preparación en administración de negocios
Preparación en soporte de tecnología
Proyectos informáticos
Consideración de mayor profundización en la preparación académica en el área de metodología de sistemas
Consideración de mayor profundización en la preparación académica en el área de ciencias de la computación
Consideración de mayor profundización en la preparación académica en el área de desarrollo de software
Calidad en la docencia del área de metodología de sistemas
Calidad en la docencia del área de ciencias de la computación
Calidad en la docencia del área de desarrollo de software
Calidad en equipamiento técnico de cursos y laboratorios

Fuente: elaboración propia.

2.10. Evaluación de resultados

De acuerdo a los resultados obtenidos en las encuestas y de las entrevistas realizadas, se hace una tabulación de datos según las opiniones dentro de cada instrumento de levantado de información. Con base en la relación entre variables y los resultado propios de cada instrumento estadístico.

La evaluación de dichos resultados se lleva a cabo en forma cualitativa y cuantitativa según sea el caso, utilizando promedios ponderados de las respuestas recabadas durante los instrumentos estadísticos.

En el caso particular de las entrevistas se evalúan los resultados por medio de un sumario de opiniones y consenso de opiniones sobre un mismo tema, para lo cual se hace referencia a la ficha de las entrevistas realizadas a egresados y a catedráticos de la carrera. Con este método se pretende obtener la opinión en común entre los entrevistados, sin dejar por un lado aquellas opiniones que no sean el común denominador, más bien estas son integradas al resumen de opiniones que se tengan respecto al tema.

En el caso de la encuesta realizada a egresados de la carrera de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala, se han obtenido los resultados presentados en dicho trabajo de investigación, sin intervenir en los mismos. Para efectos de este trabajo de investigación, únicamente se hace un análisis corto con base a dichos resultados.

3. DIAGNÓSTICO

El diagnóstico está basado en los resultados obtenidos durante la etapa de investigación de campo, por medio de la aplicación de los diferentes instrumentos planteados en el marco metodológico para la elaboración de este documento de investigación, con lo cual se pretende demostrar mediante un análisis de resultados, la validez de la hipótesis descrita al inicio. Como primer punto se realiza un análisis descriptivo; utilizando frecuencias simples y relativas, además de promedios ponderados en algunos casos. Seguido se encuentra un epígrafe en el que se sintetiza en un resumen general de resultados del diagnóstico analizando e interpretando estos desde el punto de vista del problema planteado.

3.1. Presentación y análisis de resultados

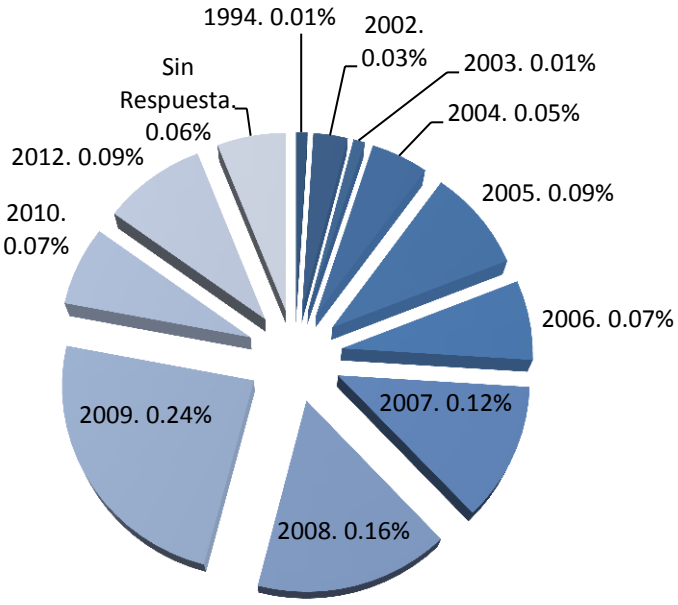
Para tener una vista panorámica del problema planteado dentro de este trabajo de investigación se presentan los resultados de las herramientas diseñadas para la captura de datos, de los cuales se hace una breve interpretación de resultados para posteriormente obtener los elementos necesarios para realizar un análisis completo y poder de esta manera determinar la veracidad de la hipótesis planteada al inicio.

3.1.1. Encuesta a recién egresados de la carrera de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala

Previo a iniciar con la presentación y análisis de resultados, se hace énfasis en que los resultados de la encuesta a recién egresados de la carrera de Ingeniería en Ciencias y Sistemas son propios de dicha encuesta y obedecen al mismo diseño de encuesta.

Dicha encuesta es la número 53963 y lleva por título: *Evaluación del pensum de estudios de la carrera de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala, respecto del perfil del egresado, año 2012.*

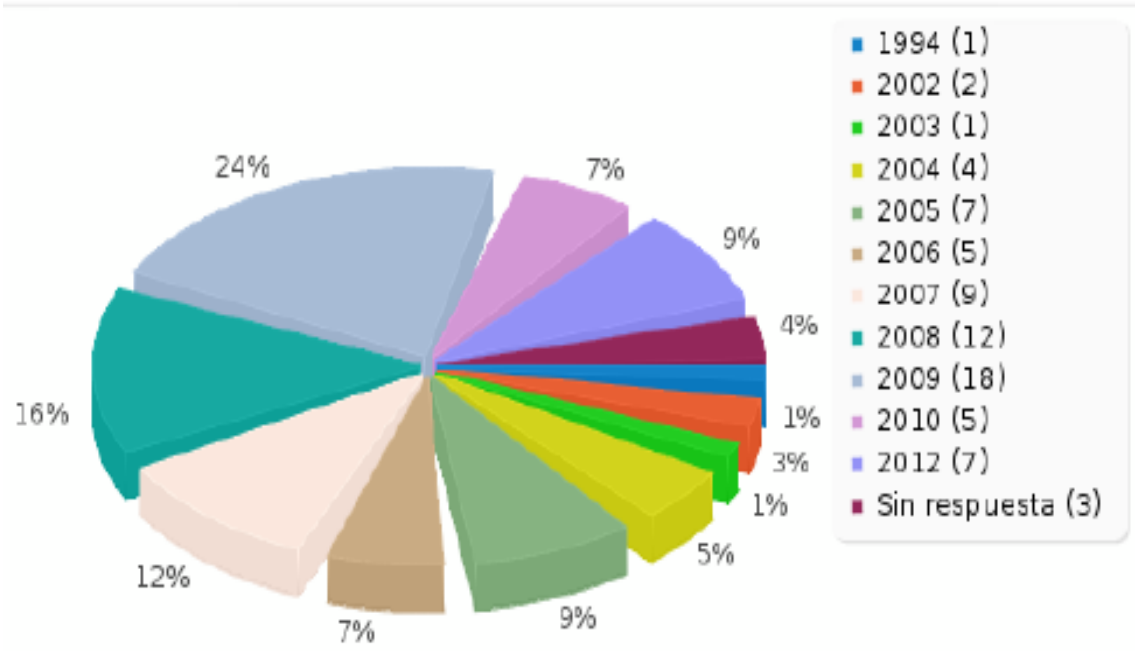
Figura 2. **Pregunta 1: ¿Año de ingreso a la Facultad de Ingeniería?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

La pregunta 1 índica la cantidad de personas que han contestado la encuesta de acuerdo al año en que ingresaron a la carrera de Ingeniería en Ciencias y Sistemas.

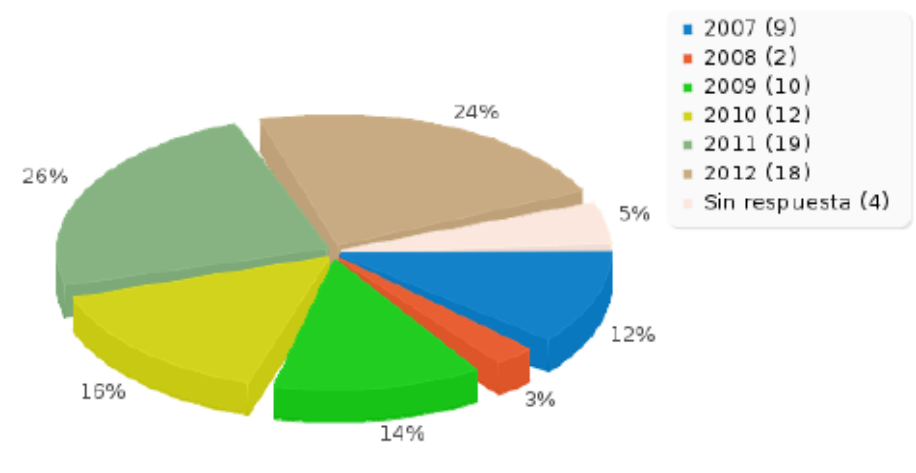
Figura 3. **Pregunta 2: ¿Año de cierre de cursos?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

Esta pregunta indica la cantidad de estudiantes que han cerrado *pensum* de estudios y el año de cierre. Es notable que la mayor cantidad de cierres se dan en el 2009, dentro de los encuestados.

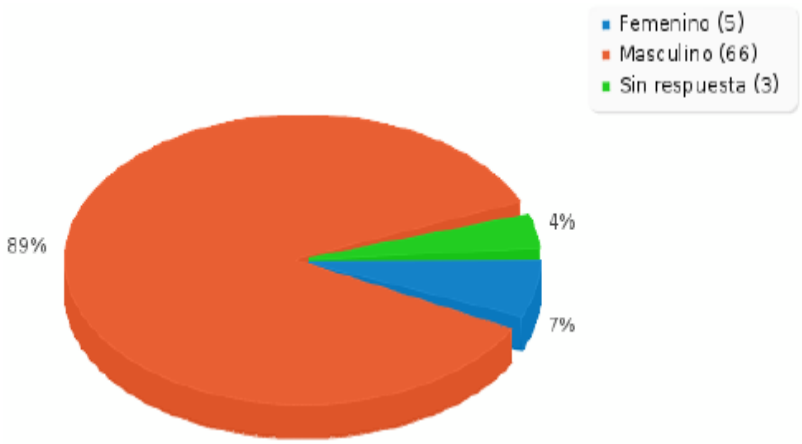
Figura 4. **Pregunta 3: ¿Año de graduación como ingeniero en Ciencias y Sistemas?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

Esta pregunta indica el año en el que se graduaron los ingenieros encuestados.

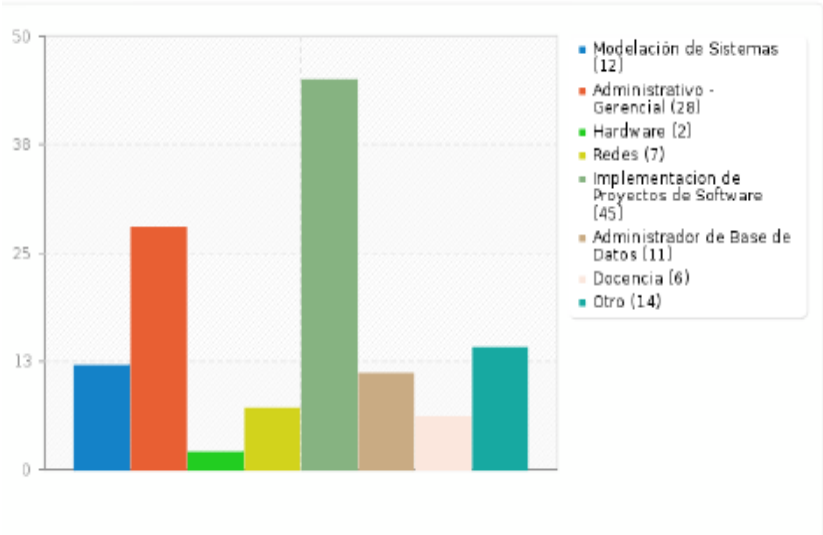
Figura 5. **Pregunta 4: ¿Sexo?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

Esta pregunta indica que la mayoría de los entrevistados son hombres y con una diferencia bastante notable, lo cual es comprensible por la cantidad de estudiantes mujeres en la carrera.

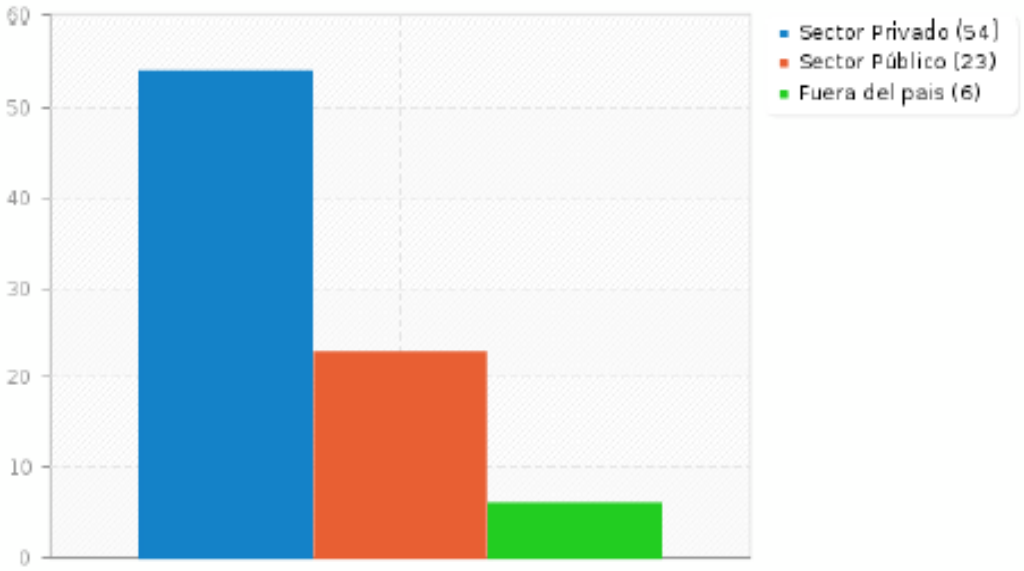
Figura 6. **Pregunta (a): ¿Hacia qué área están dirigidas tus actividades actualmente?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

Esta pregunta indica que la mayoría de los entrevistados actualmente se encuentran en 3 áreas principalmente; la primera es la de implementación de proyectos de software, la segunda es de administración y la tercera en temas de modelación de sistemas. Esto da una idea de cuáles son las áreas en las que los estudiantes de la carrera pueden tener campo para optar a un empleo, por otro lado, son áreas en las que no debe de perderse la calidad de enseñanza y mejorarse de ser posible, para fortalecer los conocimientos de los egresados de la carrera.

Figura 7. **Pregunta (b): ¿Dónde trabajas?**

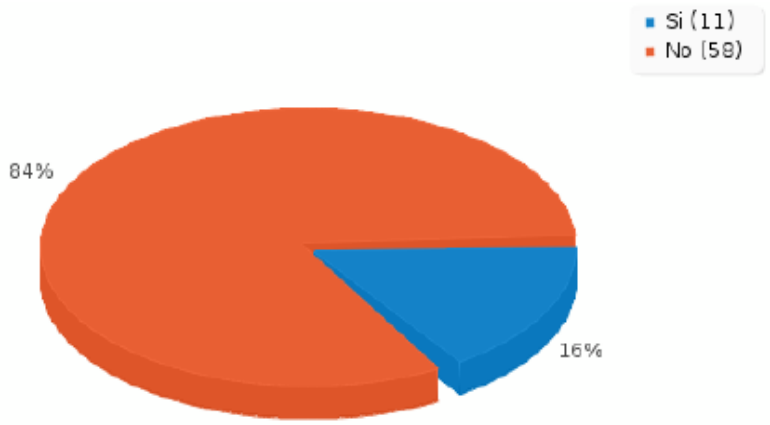


Fuente: Centro de Cálculo, Facultad de Ingeniería.

Esta pregunta da información respecto al sector en el que trabajan los encuestados, lo cual hace pensar que existe una gran aceptación de los egresados de la universidad en las empresas privadas del país.

54 se encuentran en el sector privado, 23 en el público y la minoría de 6 de los encuestados, se encuentran fuera del país.

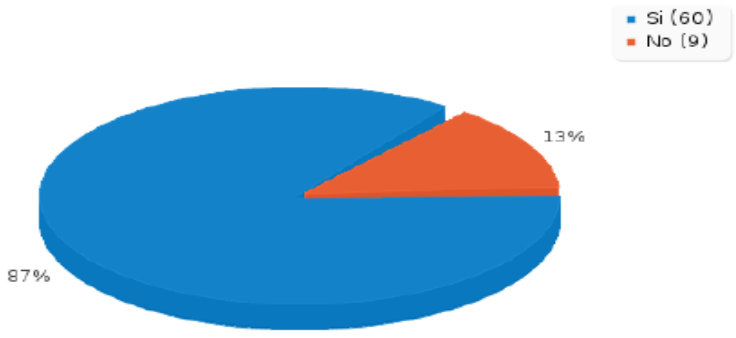
Figura 8. **Pregunta (c1): ¿El *pensum* de estudios te preparó adecuadamente para tu desempeño en administración de negocios?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

En su mayoría los encuestados afirman que el *pensum* los ha preparado para la administración de negocios.

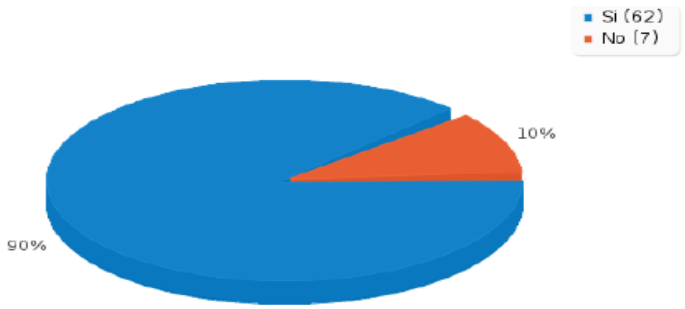
Figura 9. **Pregunta (c2): ¿El *pensum* de estudios te preparó adecuadamente para tu desempeño en soporte de tecnología?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

En su mayoría los entrevistados afirmaron haber sido preparados adecuadamente para dar soporte de tecnología

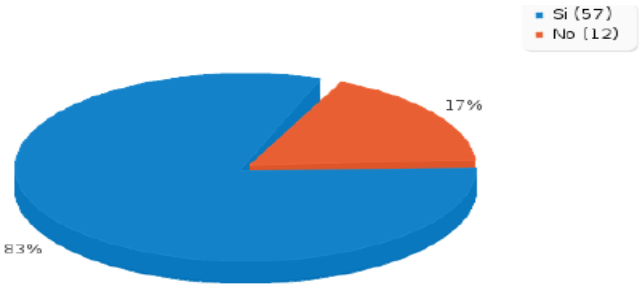
Figura 10. **Pregunta (c3): ¿El *pensum* de estudios te preparó adecuadamente para tu desempeño en proyectos informáticos?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

En su mayoría los entrevistados afirmaron haber sido preparados adecuadamente en proyectos informáticos

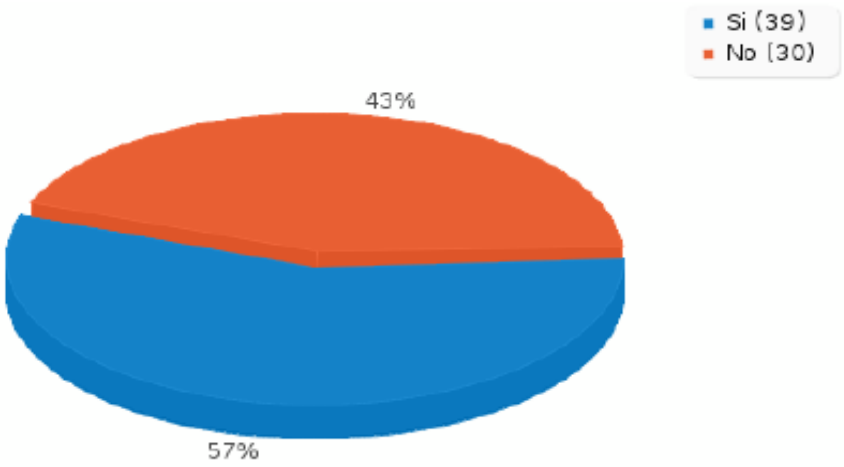
Figura 11. **Pregunta (d1): ¿Qué área consideras que necesitó mayor profundización durante tu preparación como Ingeniero en Ciencias y Sistemas (metodología de sistemas)?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

En esta pregunta los encuestados en su mayoría han expresado que si existe una deficiencia y que debería de mejorarse y profundizar la enseñanza en términos de metodologías de sistemas.

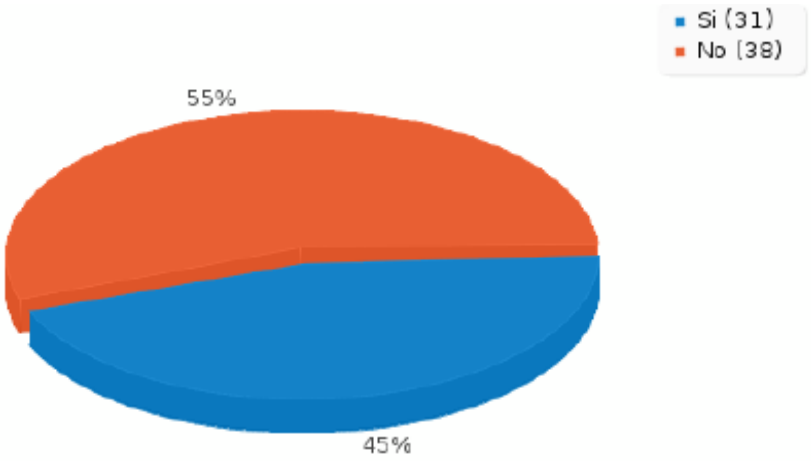
Figura 12. **Pregunta (d2): ¿Qué área consideras que necesitó mayor profundización durante tu preparación como ingeniero en Ciencias y Sistemas (ciencias de la computación)?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

En esta pregunta los encuestados aunque tienen opiniones divididas, expresan en su mayoría que existe la necesidad de mejorarse y profundizar la enseñanza de ciencias de la computación

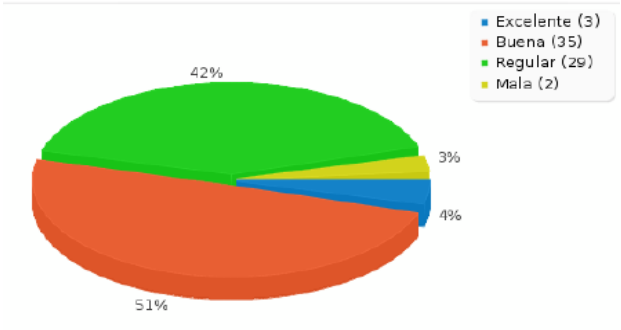
Figura 13. **Pregunta (d3): ¿Qué área consideras que necesitó mayor profundización durante tu preparación como Ingeniero en Ciencias y Sistemas (desarrollo de software)?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

En esta pregunta los encuestados en su mayoría expresan su opinión acerca de la deficiencia en la enseñanza, en términos de desarrollo de software, lo cual indica que esta área es importante y debe de mejorarse, es considerada por los encuestados una de las áreas que caracteriza a los egresados de la carrera, por lo que no debe de descuidarse y estar en continua mejora.

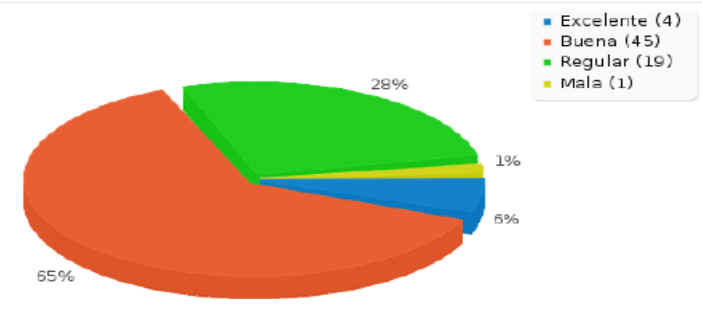
Figura 14. **Pregunta (e1): ¿Calidad en la docencia? (metodología de sistemas)**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

En su mayoría los encuestados se encuentran en que la enseñanza es buena o excelente, opinando la mayoría que es solamente buena, lo cual indica que puede mejorarse de alguna manera, un porcentaje pequeño ha opinado que es mala o regular, que de alguna manera reafirma que esta debe de mejorarse.

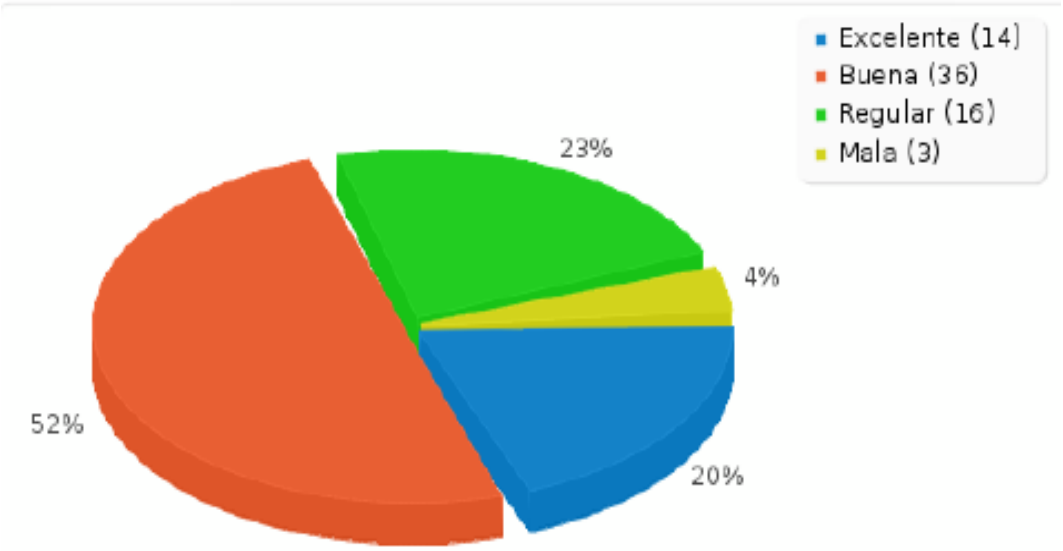
Figura 15. **Pregunta (e2): ¿Calidad en la docencia? (ciencias de la computación)**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

En su mayoría los encuestados se encuentran en que la enseñanza es buena o excelente, opinando la mayoría que es solamente buena, lo cual indica que puede mejorarse de alguna manera, un porcentaje pequeño ha opinado que es mala o regular, que de alguna manera reafirma que esta debe mejorarse.

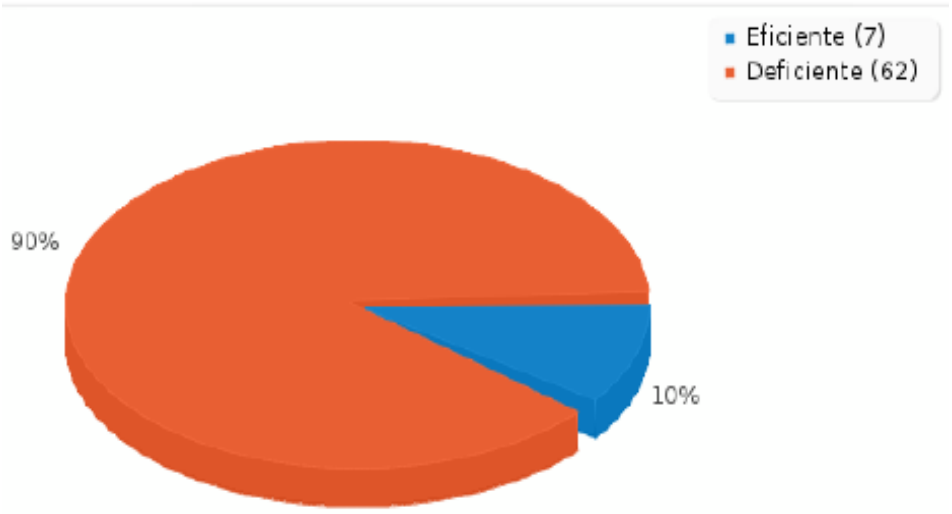
Figura 16. **Pregunta (e3): ¿Calidad en la docencia? (desarrollo de sistemas)**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

En su mayoría los encuestados se encuentran en que la enseñanza es buena o excelente, opinando la mayoría que es solamente buena, lo cual indica que puede mejorarse de alguna manera, un porcentaje pequeño ha opinado que es mala o regular, que de alguna manera reafirma que esta debe mejorarse.

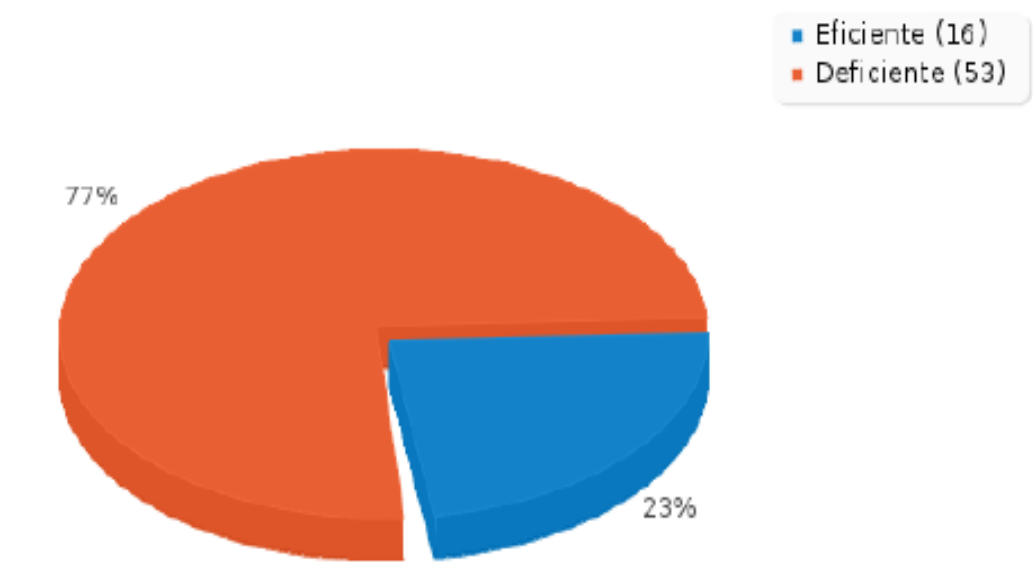
Figura 17. **Pregunta (f1): ¿Calidad de instalaciones y servicios al haber llevado los cursos y laboratorios de metodologías de sistemas, ciencias de la computación y desarrollo de software? (calidad del equipamiento técnico)**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

Esta pregunta arroja información en cuanto a la calidad, tanto de la enseñanza en los laboratorios como también que el equipamiento para la misma es deficiente, lo cual puede dar una idea de la necesidad de modernización, tanto del contenido como de la forma en la que se imparte el conocimiento.

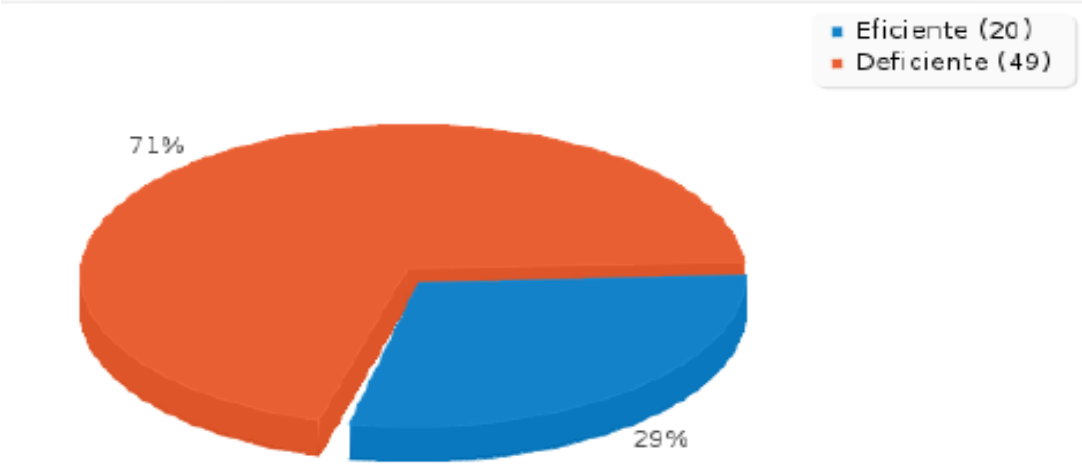
Figura 18. **Pregunta (f2): ¿Calidad de instalaciones y servicios al haber llevado los cursos y laboratorios de metodologías de sistemas, ciencias de la computación y desarrollo de software? (calidad en las instalaciones)**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

Esta pregunta brinda información en cuanto a la calidad de las instalaciones para impartir el conocimiento establecido en el *pensum* de la carrera, a lo cual la mayoría de los encuestados coinciden en que no es el más eficiente.

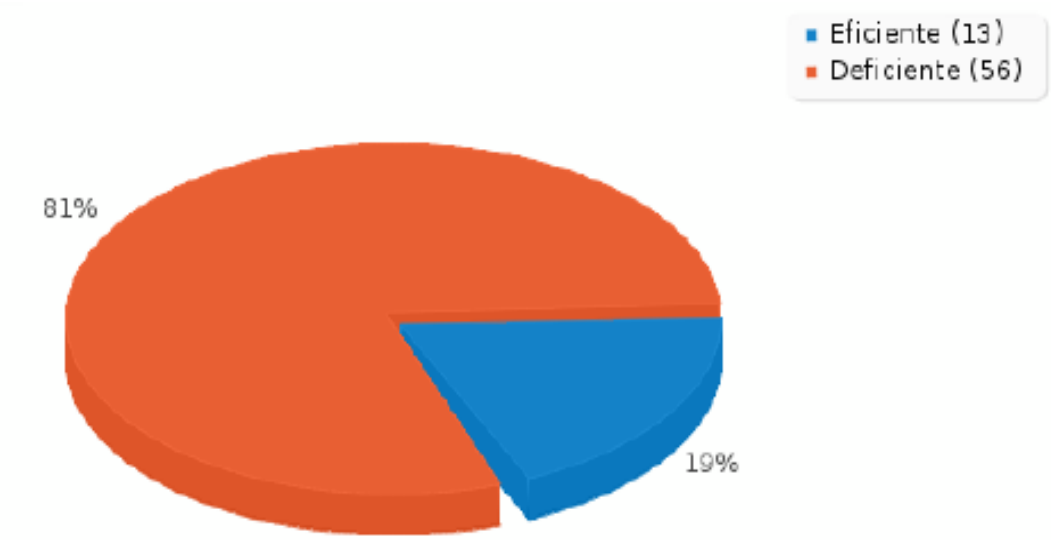
Figura 19. **Pregunta (f3): ¿Calidad de instalaciones y servicios al haber llevado los cursos y laboratorios de metodologías de sistemas, ciencias de la computación y desarrollo de software? (atención al estudiante por parte de catedráticos)**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

En cuanto a esta pregunta, el resultado indica que existe una falta de atención de los catedráticos hacia los alumnos, es decir existe una carencia en este sentido, lo cual es un factor que puede llegar a ser importante en el aprendizaje de los temas de la carrera; para lo cual debe de implementarse mecanismos de control.

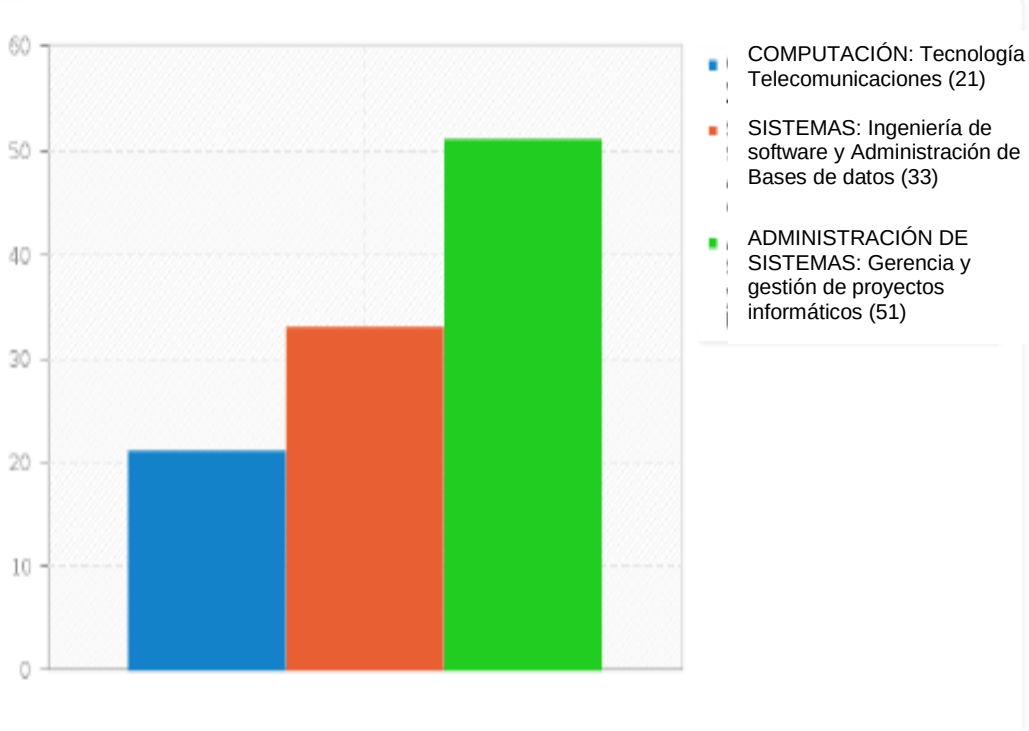
Figura 20. **Pregunta (f4): ¿Calidad de instalaciones y servicios al haber llevado los cursos y laboratorios de metodologías de sistemas, ciencias de la computación y desarrollo de software? (atención al estudiante por parte de auxiliares)**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

Esta pregunta refleja la realidad de la mala atención que reciben los estudiantes por parte de los asistentes de cátedra, lo cual es un factor que debe de mejorarse y buscar mejores mecanismos de control

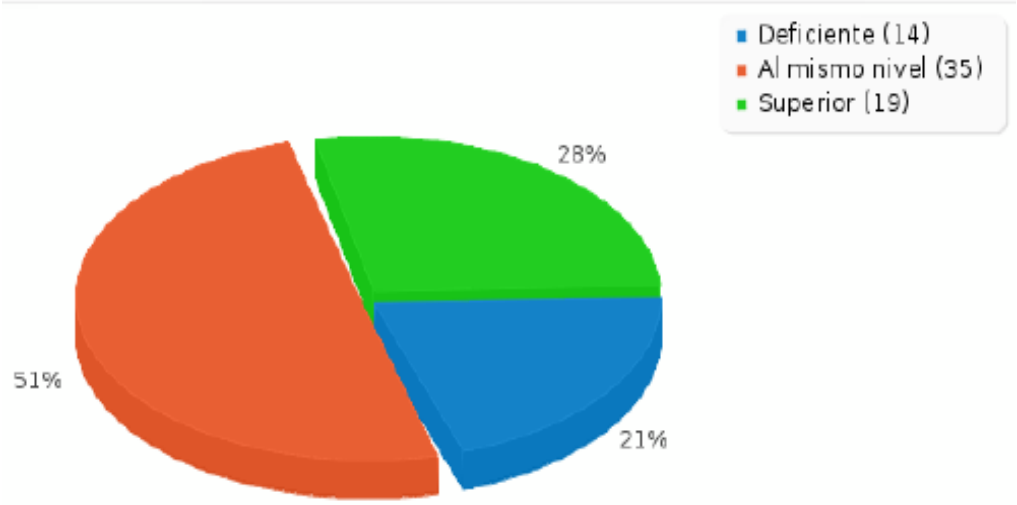
Figura 21. **Pregunta (g): de las tres especialidades propuestas a continuación para preparar a los futuros ingenieros en Ciencias y Sistemas, ¿cuál escogería?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

Los resultados de esta pregunta indican la preferencia por una especialización en la administración de sistemas, gerencia y gestión de proyectos informáticos

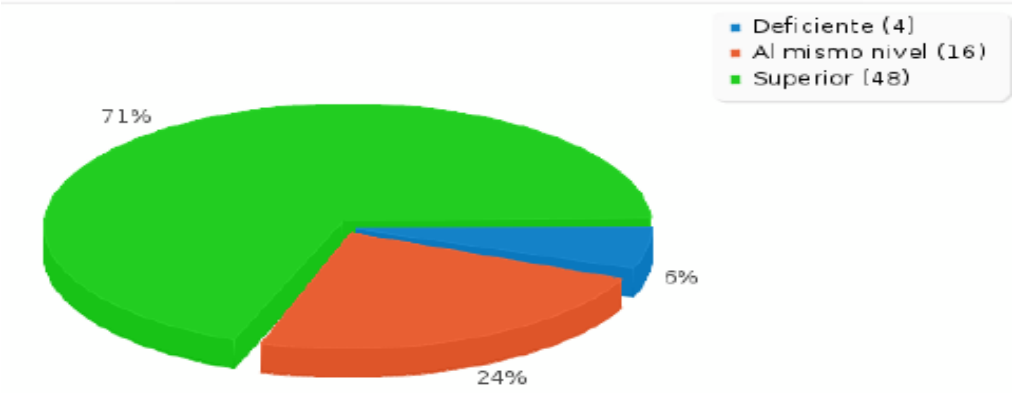
Figura 22. **Pregunta (h1): ¿Cómo comparas al ingeniero egresado de la USAC con respecto al egresado de otra universidad (en conocimientos de tecnología y telecomunicaciones con respecto a universidades privadas)?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

Los resultados de esta pregunta apuntan a que los estudiantes consideran estar en su mayoría al mismo nivel de conocimientos de tecnología y telecomunicaciones con respecto a las universidades privadas, siendo la segunda respuesta más alta que se encuentran a un nivel superior comparado con las otras universidades en relación al mismo tema. Con un 21 por ciento indicando que se está por debajo del nivel de los egresados de las universidades privadas.

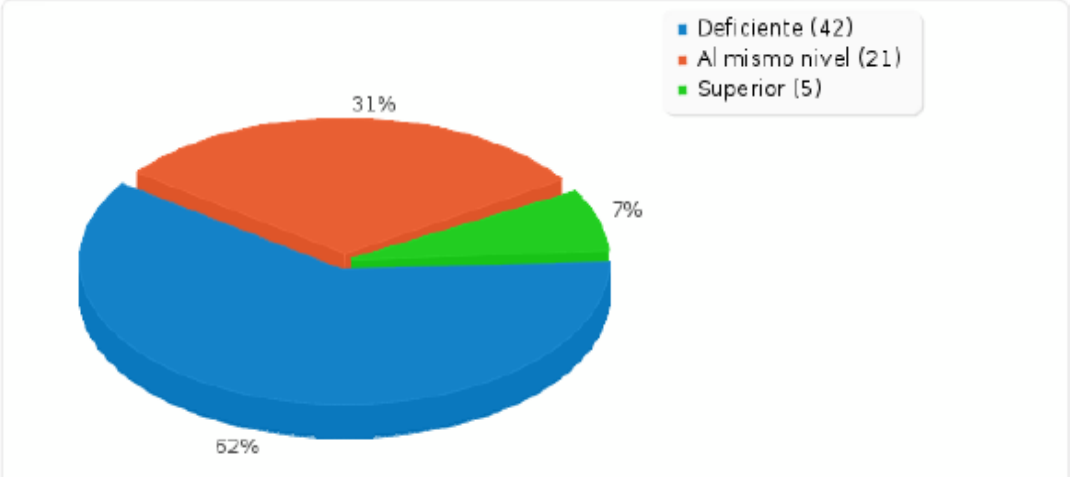
Figura 23. **Pregunta (h2): ¿Cómo comparas al ingeniero egresado de la USAC con respecto al egresado de otra universidad (en conocimientos de ingeniería de software con respecto a universidades privadas)?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

En esta pregunta parece haber una mayoría notable que opina que se encuentran por encima del nivel de conocimientos de ingeniería de software, comparado con otras universidades privadas y tan solo un 20 por ciento opina que se encuentra a un mismo nivel que otras universidades y un nivel muy pequeño que se encuentra por debajo del nivel de las universidades privadas.

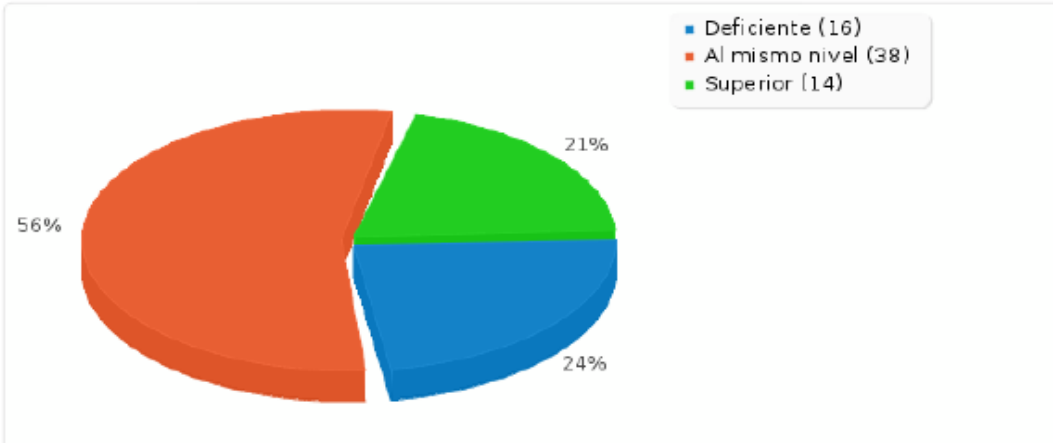
Figura 24. **Pregunta (h3): ¿Cómo comparas al ingeniero egresado de la USAC con respecto al egresado de otra universidad (en conocimientos de administración de sistemas con respecto a universidades privadas)?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

En esta pregunta la gran mayoría coincide que los conocimientos de administración de sistemas y gestión de proyectos están por debajo de los niveles de universidades privadas en el país, lo cual puede interpretarse como una contradicción clara con la pregunta C1, entre otras en las que se afirma que se adquieren los conocimientos suficientes para desenvolverse dentro del área de administración y gestión de proyectos. Y coincide con que esta es una de las áreas que debe de fortalecerse dentro de la carrera.

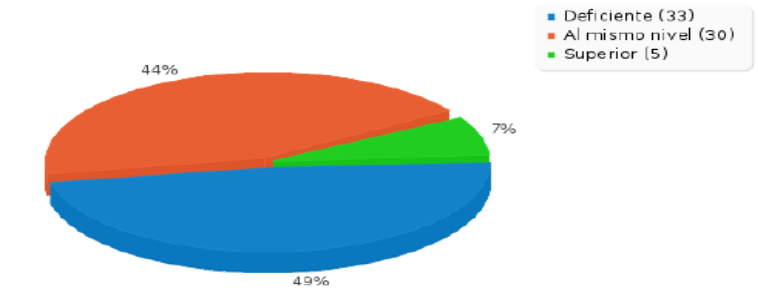
Figura 25. **Pregunta (h4): ¿Cómo comparas al ingeniero egresado de la USAC con respecto al egresado de otra universidad (en conocimientos generales con respecto a una universidad centroamericana)?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

Los resultados de esta pregunta indican que los egresados de la carrera de Ingeniería en Ciencias y Sistemas se sienten en desventaja en cuanto a conocimientos generales comparados con universidades de Centro América.

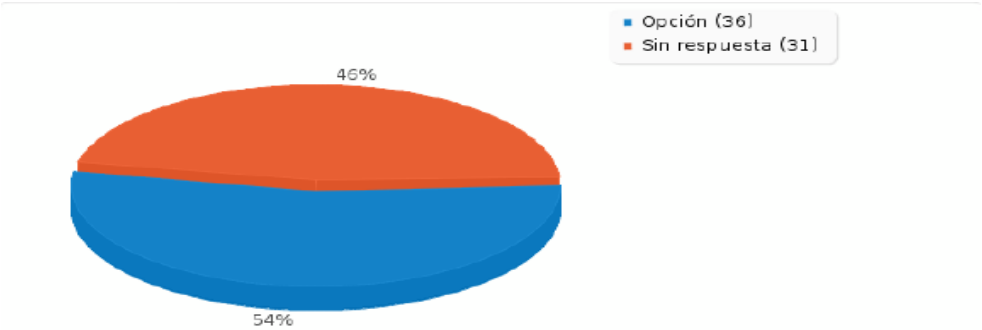
Figura 26. **Pregunta (h5): ¿Cómo comparas al ingeniero egresado de la USAC con respecto al egresado de otra universidad (en conocimientos generales con respecto a universidades de Latinoamérica)?**



Fuente: Centro de Cálculo, Facultad de Ingeniería.

Ampliando un poco más la perspectiva, los resultados de esta pregunta indican en un 49 por ciento que los estudiantes de la carrera se encuentran a nivel menor en conocimientos generales comparados con los estudiantes de universidades latinoamericanas.

Figura 27. **Pregunta i): ¿Tienes algo que comentar respecto a la calidad del egresado de la escuela?**



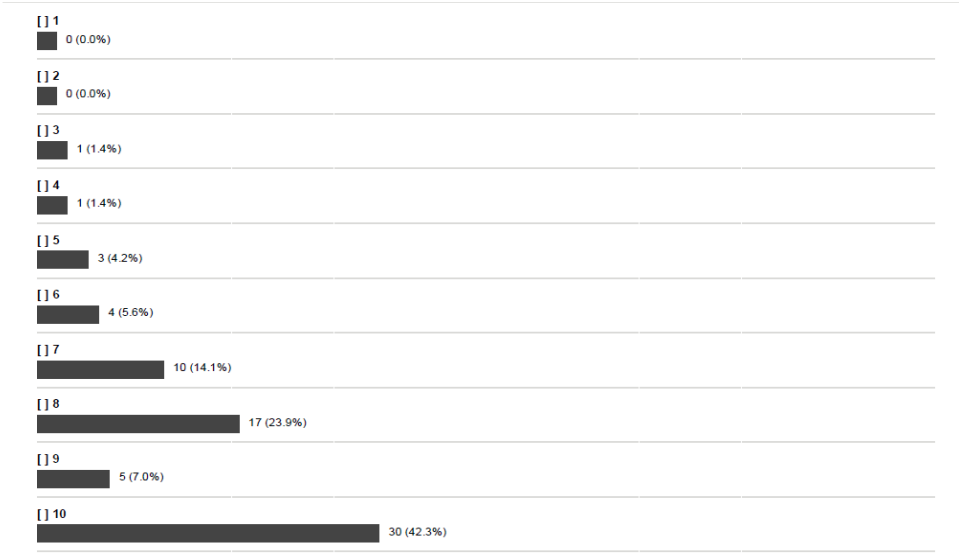
Fuente: Centro de Cálculo, Facultad de Ingeniería.

Respecto a esta pregunta la mayoría de los encuestados coinciden en que si hay un comentario que hacer y los otros 46 por ciento coinciden en que no hay nada que comentar.

3.1.2. Encuesta a estudiantes

Esta encuesta ha sido realizada específicamente para la elaboración de este trabajo de investigación durante el primer semestre del 2013, a los estudiantes de la carrera de Ingeniería en Ciencias y Sistemas, de acuerdo al diseño de muestra especificado en el marco metodológico.

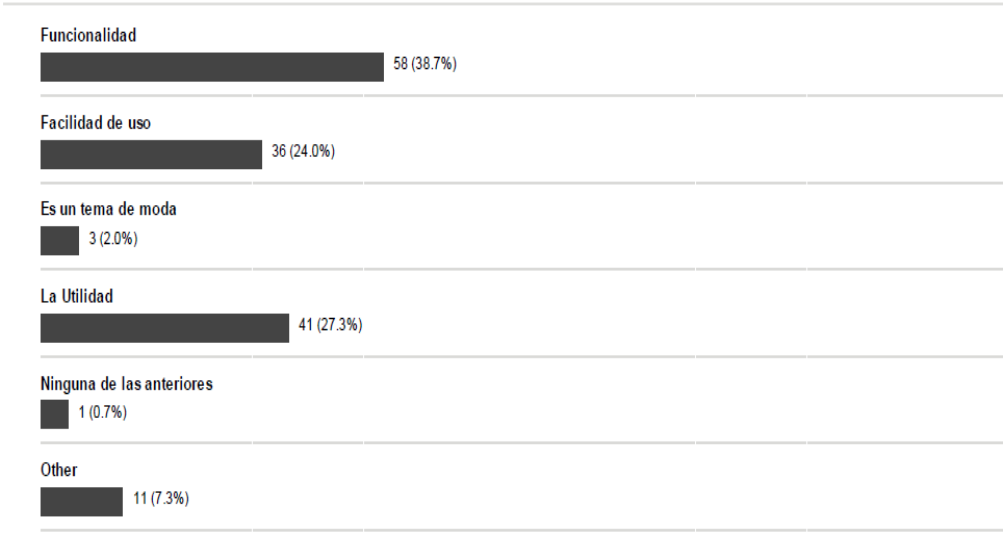
Figura 28. **Pregunta 1: en la escala de 1 a 10, donde 10 es muy interesante y 1 es nada interesante, ¿qué tan interesante le parece el tema: las buenas prácticas de desarrollo de software?**



Fuente: elaboración propia.

Esta pregunta denota la conciencia de los alumnos respecto a la importancia que tiene las buenas prácticas en el desarrollo de software, indicando que la mayoría con un 42 por ciento está muy interesada en aprender este tipo de temas, y otra gran mayoría está bastante interesada en aprender, dándole a la escala un 8 con un 23 por ciento, y así disminuye el porcentaje, teniendo un 0 por ciento de los encuestados que opinan que no es importante.

Figura 29. **Pregunta 2: ¿Cuál o cuáles de las siguientes características le atraen del tema de implementación de buenas prácticas de desarrollo de software?**



Fuente: elaboración propia.

En esta pregunta la mayoría de los encuestados se ha inclinado por la funcionalidad con un 38 por ciento y un 27 por ciento se inclinan por la utilidad, mientras otro 24 por ciento se inclina por la facilidad de uso.

Figura 30. **Pregunta 3: ¿En qué cursos de la carrera le han enseñado de buenas prácticas para el desarrollo de software?**

	Nada	Poco	Regular	Mucho	SNE	Responses	Average Score
IPC1 e IPC2	**31 (43.66%)**	27 (38.03%)	10 (14.08%)	3 (4.23%)	0 (0.00%)	71	1.79 / 5(35.80%)
Manejo e implementación de Archivos	**36 (50.70%)**	28 (39.44%)	7 (9.86%)	0 (0.00%)	0 (0.00%)	71	1.59 / 5(31.80%)
Estructuras de Datos	**27 (38.03%)**	17 (23.94%)	15 (21.13%)	12 (16.90%)	0 (0.00%)	71	2.17 / 5(43.40%)
Sistemas de Bases de Datos 1 y Sistemas de Bases de Datos2	**21 (29.58%)**	19 (26.76%)	**21 (29.58%)**	9 (12.68%)	1 (1.41%)	71	2.30 / 5(46.00%)
AyD1 y AyD2	1 (1.41%)	7 (9.86%)	29 (40.85%)	**31 (43.66%)**	3 (4.23%)	71	3.39 / 5(67.80%)
Seminario de Sistemas 1 y Seminario de Sistemas 2	**27 (38.03%)**	22 (30.99%)	16 (22.54%)	2 (2.82%)	4 (5.63%)	71	2.07 / 5(41.40%)
Software Avanzado	4 (5.63%)	13 (18.31%)	12 (16.90%)	9 (12.68%)	**33 (46.48%)**	71	3.76 / 5(75.20%)
							2.44 / 5 **(48.77%)**

Fuente: elaboración propia.

En esta pregunta los encuestados tienen distintas opiniones, pero los resultados apuntan a que han recibido mayor contenido en relación a buenas prácticas en los cursos de Análisis y Diseño de Sistemas 1 y 2, y en el curso de Software Avanzado. Además de eso, se puede observar que un 43,66 por ciento el porcentaje más alto ha respondido que ha obtenido mayor conocimiento acerca de buenas prácticas en los cursos de Análisis y Diseño de Sistemas 1 y 2.

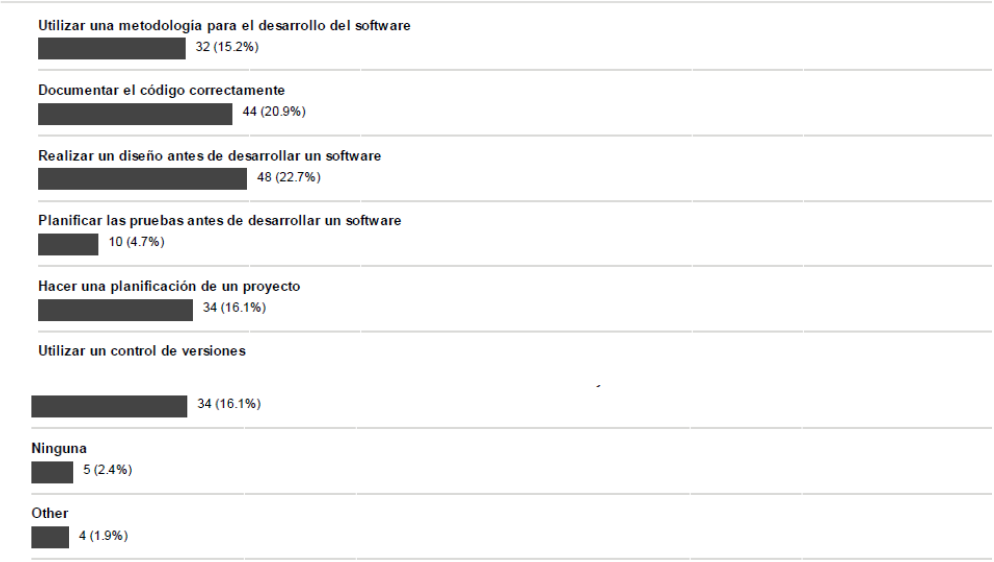
Figura 31. **Pregunta 4: ¿En qué cursos se debería de reforzar la enseñanza de buenas prácticas de desarrollo de software?**

	Nada	Poco	Regular	Mucho	SNE	Responses	Average Score
IPC1 e IPC2	1 (1.41%)	3 (4.23%)	13 (18.31%)	**54 (76.06%)**	0 (0.00%)	71	3.69 / 5(73.80%)
Manejo e implementación de Archivos	5 (7.04%)	13 (18.31%)	**29 (40.85%)**	24 (33.80%)	0 (0.00%)	71	3.01 / 5(60.20%)
Estructuras de Datos	5 (7.04%)	14 (19.72%)	**27 (38.03%)**	24 (33.80%)	1 (1.41%)	71	3.03 / 5(60.60%)
Sistemas de Bases de Datos 1 y Sistemas de Bases de Datos2	5 (7.04%)	12 (16.90%)	**32 (45.07%)**	22 (30.99%)	0 (0.00%)	71	3.00 / 5(60.00%)
AyD1 y AyD2	5 (7.04%)	3 (4.23%)	15 (21.13%)	**47 (66.20%)**	1 (1.41%)	71	3.51 / 5(70.20%)
Seminario de Sistemas 1 y Seminario de Sistemas 2	3 (4.23%)	11 (15.49%)	**37 (52.11%)**	18 (25.35%)	2 (2.82%)	71	3.07 / 5(61.40%)
Software Avanzado	0 (0.00%)	5 (7.04%)	13 (18.31%)	**38 (53.52%)**	15 (21.13%)	71	3.89 / 5(77.80%)
							3.31 / 5 (66.29%)

Fuente: elaboración propia.

De los encuestados, un 76 por ciento indica que debe reforzarse mucho en los cursos de Programación y Computación 1 y 2, otros opinan que debería de reforzarse mucho también en Análisis y Diseño de Sistemas con un 66 por ciento, y un 53 por ciento opina que debe de reforzarse mucho también en el área de Software avanzado, asimismo en los cursos de Bases de Datos, también opina un 32 por ciento que debería de reforzarse regularmente este curso con buenas prácticas de desarrollo de sistemas.

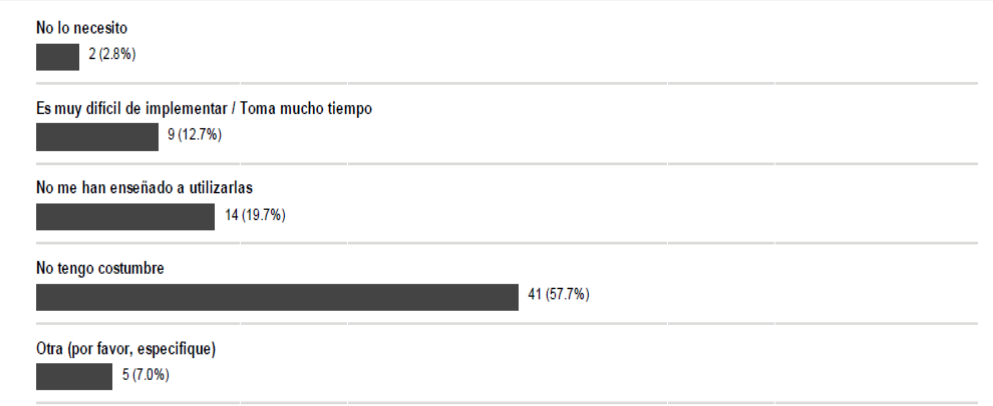
Figura 32. **Pregunta 5: De las siguientes buenas prácticas de desarrollo de sistemas, ¿cuáles son las que comúnmente utiliza para el desarrollo de software?**



Fuente: elaboración propia.

En esta pregunta la mayoría coincide que realiza un diseño del sistema con un 48 por ciento, otros con un 44 por ciento coinciden que documentan el código del software, y un 32 por ciento utiliza una metodología de desarrollo de software, un 34 por ciento realiza una planificación del proyecto, también cabe mencionar que el 34 por ciento utiliza un sistema de versiones. Cabe mencionar que existe un 2,4 por ciento que no aplica ninguna de estas prácticas, y otro porcentaje también pequeño 4,7 por ciento Planifica las pruebas a su sistema.

Figura 33. **Pregunta 6: ¿Cuál o cuáles de las siguientes características reflejan de alguna manera su realidad en cuanto a la implementación de las buenas prácticas de desarrollo de software?**



Fuente: elaboración propia.

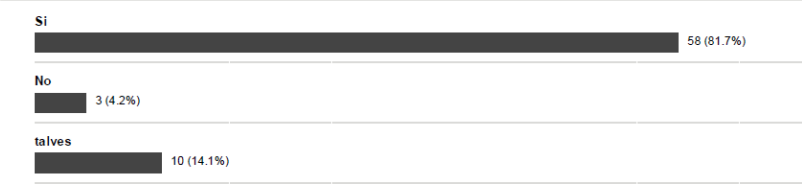
En esta pregunta los estudiantes encuestados se inclinaron en un 57 por ciento por la respuesta en la que expresan que no poseen la costumbre de utilizar buenas prácticas de desarrollo de software, lo cual indica que existe la conciencia de la necesidad que representan estas, pero que se utiliza un desarrollo viciado, en el cual no incluyen buenas prácticas por desconocimiento (un 19%) y un 12 por ciento indica que le es difícil aplicarlas, siendo en su mayoría los estudiantes que indican que no tienen la costumbre de utilizarlas.

Figura 34. **Pregunta 7: ¿Cuál o cuáles de las siguientes prácticas le parece importante utilizar para el desarrollo de software?**

Fuente: elaboración propia.

En la pregunta 7, la opinión de los estudiantes encuestados está dividida en todas las buenas prácticas propuestas, teniendo una leve mayoría el análisis de requerimientos, seguido por la planificación de las actividades. Con el mismo porcentaje (10%), se encuentran las buenas prácticas que consisten en: definición de una arquitectura acorde, comenzar a hacer código solo hasta haber entendido bien el diseño general del sistema, automatización del proceso de versionamiento.

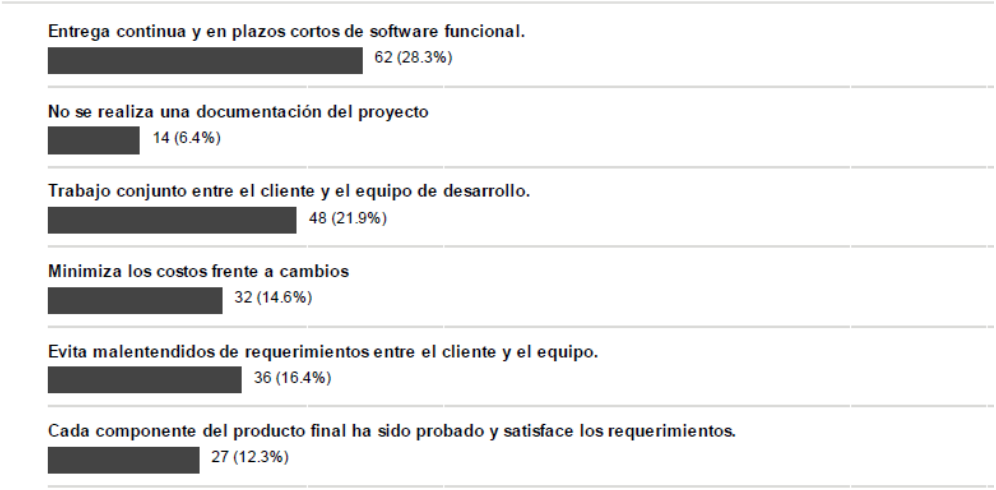
Figura 35. **Pregunta 8: ¿Creé que la aplicación integrada de las prácticas, en el contexto de una metodología de desarrollo de software, debe de ser parte del contenido y prácticas de los cursos de desarrollo de software?**



Fuente: elaboración propia.

La opinión de los encuestados en relación a la pregunta 8, es que si es necesario con un 8,7 por ciento afirmando, que las buenas practicas dentro del marco de una metodología deben ser integrarlas en los cursos de desarrollo de software; y tan solo un 4,2 por ciento teniendo una opinión negativa respecto a la aplicación integrada de buenas prácticas, y un 14,1 por ciento indicando un tal vez, lo cual puede deberse al desconocimiento propiamente de las mismas.

Figura 36. **Pregunta 9: ¿Cuáles de las siguientes características creé usted que describe una metodología ágil?**



Fuente: elaboración propia.

El objetivo de la pregunta 9 es identificar el conocimiento relacionado al tema de metodologías ágiles, ya que es en gran parte el tipo de metodologías que se imparte durante la carrera de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala.

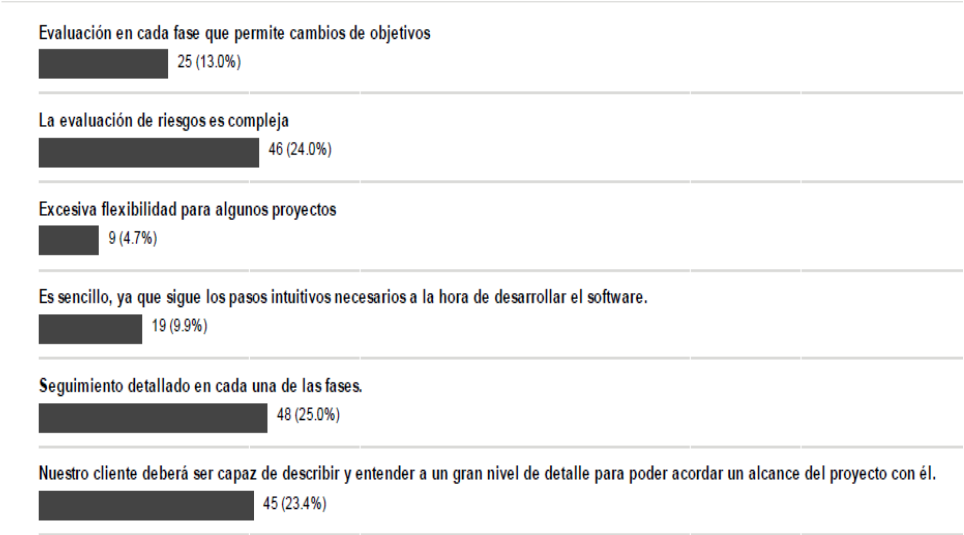
Las respuestas acertadas son la mayoría, pero existe por ejemplo; 6,4 por ciento que indica que este tipo de metodologías no realiza una documentación, lo cual no es cierto, ya que es muy importante que se realice una documentación que describa tanto el proyecto y que pueda servir más adelante para futuros usuarios y personas que deseen modificar dicho proyecto.

Otro porcentaje de 14,5 por ciento indica que minimiza los costos frente a cambios, y esta no representa una característica como tal de este tipo de

metodologías, ya que en algunas como por ejemplo XP, esta representa precisamente una de sus debilidades.

Siguiendo con el análisis, se ve que en la mayoría de los casos, los estudiantes conocen las características y ventajas de este tipo de metodologías de desarrollo.

Figura 37. **Pregunta 10: ¿Cuáles de estas características corresponden a una metodología rígida o tradicional?**



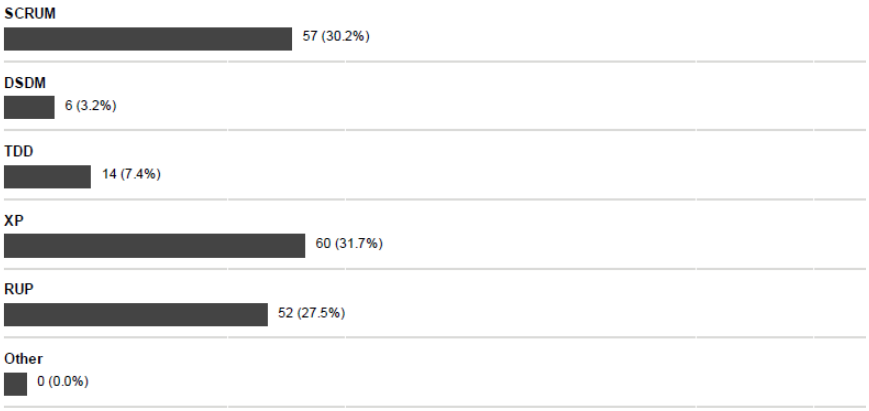
Fuente: elaboración propia.

La mayoría de los estudiantes encuestados opina que la principal característica es el seguimiento detallado en cada una de las fases, lo cual es correcto, pero esta respuesta corresponde únicamente al 25 por ciento de los encuestados, otro 23 por ciento corresponde a que el cliente (en este caso el evaluador del proyecto deberá de ser capaz de describir y entender a un gran nivel de detalle para poder acordar un alcance del proyecto con él.

Los estudiantes opinan que existe una excesiva flexibilidad para algunos proyectos, lo cual no concuerda con una metodología rígida aplicada, y permite mayores variantes (4,7 %). También el 9 por ciento de los encuestados opina que es sencillo y que se siguen pasos intuitivos a la hora de desarrollar el software, lo cual no es característica de este tipo de metodologías, ya que se establece y diseña previamente para poder desarrollar el software.

Por lo general se puede decir que los estudiantes, no conocen bien este tipo de metodologías.

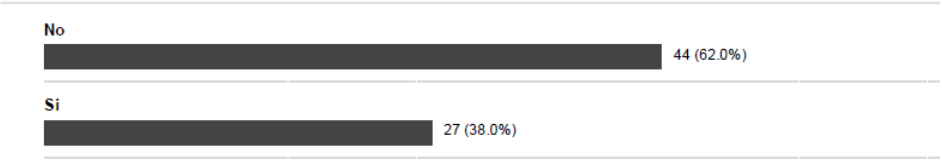
Figura 38. **Pregunta 11: ¿Qué metodología(s) de desarrollo de software considera usted que deben de enseñarse y aplicarse en los distintos años de la carrera?**



Fuente: elaboración propia.

En esta pregunta es evidente que los alumnos se inclinan más por metodologías de desarrollo como Scrum, XP, y RUP, esto es posible que se deba a que las conocen, y desconocen otras. Nadie indicó otra metodología de desarrollo diferente a las propuestas.

Figura 39. **Pregunta 12: ¿Tiene algún comentario o sugerencia para mejorar la calidad del software que se realiza durante los diferentes años de la carrera?**



Fuente: elaboración propia.

La mayoría de los estudiantes expreso no tener ningún comentario al respecto, un porcentaje considerable (38%) tuvo las siguientes opiniones en general:

- Debe de enseñarse desde el principio los temas versionamiento de sistemas.
- Debe enseñarse de mejor manera y más ampliamente a aplicar las metodologías de desarrollo de sistemas.
- Se debería de enseñar mejor la administración de proyectos
- Se debería de realizar menos proyectos en grupo, ya que es difícil coordinarse con los compañeros.

3.2. Interpretación y análisis por *ítem*

En cada uno de los *ítems* propuestos, se obtiene información valiosa, con la cual se conoce una realidad del proceso de desarrollo de software, y de acuerdo a esta, se identifican puntos clave dentro del proceso que deben abordarse de manera efectiva y estratégica.

3.2.1. Análisis de contenido los cursos

La Ingeniería de Sistemas está relacionada con todos los aspectos asociados al desarrollo de sistemas complejos: hardware, software, y otros. Los sistemas intensivos de software son sistemas constituidos principalmente por software. Durante la carrera de Ingeniería en Ciencias y Sistemas, los alumnos de la misma adquieren conocimientos relacionados al proceso de ingeniería de software.

Al estudiar los cursos del área de desarrollo de software, se encuentran los siguientes contenidos:

- Abstracción de objetos y uso de UML
- Programación orientada a objetos
- Estructuración de datos
- Arquitecturas de software
- Aplicación de metodologías de desarrollo de software.
- Análisis y diseño de software
- Formulación y gestión de proyectos de software
- Entre otros

Para este análisis se ha realizado una ficha resumen de los cursos del área de desarrollo de software, y así poder ampliar el conocimiento en relación al contenido y el proceso de desarrollo como conocimiento adquirido durante el transcurso de la carrera, estos cursos son:

- Introducción a la programación y computación 1 y 2
- Manejo e implementación de archivos
- Estructuras de datos

- Sistemas de Bases de Datos 1 y 2
- Seminario de sistemas 1 y 2
- Análisis y diseño de sistemas 1 y 2.
- Software avanzado

De acuerdo al resultado de este análisis, se observa un vago y no muy estructurado proceso de desarrollo, en el cual las metodologías de desarrollo son uno de los últimos temas de aprendizaje e implementación; las consecuencias de esto derivan en la mayoría de los casos una aplicación deficiente de las metodologías.

Es el mismo caso con el tema de la calidad del software, su aplicación es deficiente y un aspecto muy importante es considerar que existe un proceso de aprendizaje para el desarrollo de proyectos de software, y representa un doble esfuerzo para los estudiantes adquirir nuevas estrategias para lograr una mejor calidad de software, así como un proceso más eficiente, se refiere a lo difícil que es integrar nuevas estrategias de diseño y buenas prácticas dentro del Proceso de Desarrollo de Software (PDS) cuando ya se está acostumbrado a un proceso de desarrollo con ciertos vicios y aceptación para un margen de error un poco alto tanto en el proceso de desarrollo como en el producto final.

3.2.2. El proceso de desarrollo de software actualmente

Para visualizar el PDS que actualmente siguen los estudiantes de Ingeniería en Ciencias y Sistemas, se detallan las actividades más comunes dentro de dicho proceso, utilizando la siguiente tabla resumen.

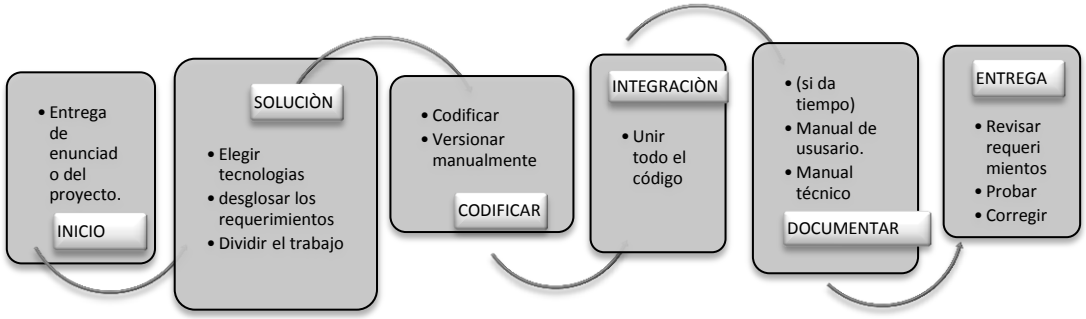
Tabla VIII. **Resumen del PDS actual**

Toma de requerimientos	• Se adquiere el conocimiento para la identificación de los requerimientos y su priorización, en los últimos cursos del área y no en todos los semestres, por lo que es evidente que se adquiere de manera tardía.
Análisis y diseño	• Este tema es abordado en los últimos cursos del área y el contenido no es estandard en todos los semestres.
Construcción del software	• El contenido respecto al control del desarrollo del proyecto es muy poco • El tema de administración de la configuración se adquiere en los últimos cursos del área, antes de elo no existe ninguna referencia. • El contenido respecto al diseño se concentra al final de la carrera.
Pruebas del software	• El contenido y aplicación del tema de pruebas es abordado de forma corta en los cursos de Análisis y diseño, por contemplarse dentro de la integración continua. • No existe el tema de diseño de pruebas, al menos de manera formal y que se aborde todos los semestres.
Integración del software	• El tema de la planificación de integración es vagamente abordado en algunos semestres
Mantenimiento del software	• El mantenimiento no está contemplado dentro del contenido, mas que el control de cambios de manera escueta y corta.

Fuente: elaboración propia

El proceso de desarrollo de software que sigue la mayoría de estudiantes de Ingeniería en Ciencias y Sistemas, es bastante simple; las actividades básicas definidas dentro de estos se reflejan en el siguiente diagrama:

Figura 40. **PDS actual en la línea del tiempo**



Fuente: elaboración propia

Como se observa en el diagrama, existen 6 etapas en el desarrollo de un proyecto de software durante los cursos de la carrera, y estos han sido adoptados por los estudiantes de forma natural, con base en la necesidad y las circunstancias; es decir las exigencias de los cursos y el tiempo que poseen para desarrollar un proyecto de software.

- La primera, consiste en la entrega del enunciado del proyecto, con el problema a resolver. En la mayoría de los casos el auxiliar de la cátedra o el catedrático del curso, dan una explicación del problema y resuelven dudas acerca del enunciado.

- La segunda etapa consiste en la organización de la solución, es decir en grupo o individualmente, los estudiantes se organizan, con lo que se ha entendido, deciden usar una tecnología u otra; esto con base en la experiencia con la tecnología, o bien si es recomendada o requerida en el enunciado del proyecto. Luego de ello se procede a desglosar el

proyecto, dividir el sistema en pequeñas partes y cada integrante se compromete a desarrollar, es decir codificar la parte que le corresponda.

- La tercera etapa es la de codificación, desarrollar el software según lo que se entendió en el enunciado. Durante esta etapa en la mayoría de los casos se va avanzando en el desarrollo de acuerdo a lo que se va leyendo en el enunciado. Conforme a lo que se va leyendo, se va avanzando; puede sonar un poco complicado pero esta práctica es muy común, se decide iniciar el proyecto y se leen las primeras líneas y de acuerdo a lo que en estas se solicite se inicia, cuando se ha terminado o avanzado en esto, se lee quizás otro párrafo y se procede a codificar.

- La cuarta etapa, consiste en unir todos los pequeños sistemas desarrollados con las diferentes funcionalidades para conformar el producto final. En algunos casos se dan pequeñas integraciones a lo largo del desarrollo cuando es posible.

- La quinta etapa, si aún se tiene un poco de tiempo, se desarrolla la documentación del proyecto, en la mayoría de los casos es una descripción de los *screenshots* del sistema final, y explicando la manera en la que opera el sistema. Esta documentación carece de profundidad, así como de bases que permitan a un usuario final entender o aclarar dudas en su totalidad.

- La sexta etapa se da durante el período de entrega; durante este el auxiliar de cátedra o bien el catedrático revisan conjuntamente con el estudiante los requisitos del sistema, para verificar que estos hayan sido realizados completamente. Es común que el sistema presente errores,

por lo que los estudiantes tienen que corregirlos si el catedrático o asistente de cátedra lo autoriza y si es posible corregir en poco tiempo.

Es importante mencionar que existen en algunos casos, variantes dentro del PDS descrito anteriormente; estas variantes dependen de la organización de los estudiantes y en muchos casos de la complejidad del proyecto, por ejemplo:

Tabla IX. **Variantes del PDS**

Variantes	Priorización de los requerimientos.
	Tomar screenshots del proceso, para luego editar el manual.
	Integraciones cuando una parte del sistema ya está terminado
	Probar manualmente los modulos que se han terminado
	Revisar los requisitos de entrega, si estos ya se conocen

Fuente: elaboración propia

3.2.3. Encuestas a los estudiantes de la carrera

Las encuestas realizadas durante la presente investigación, fueron aplicadas a los estudiantes de los últimos cursos del área de desarrollo de software, de acuerdo al diseño muestral. Se consideró los últimos cursos del área de desarrollo de software, ya que los conocimientos del tema se espera sean uniformes entre la población estudiantil, es decir serán a un mismo nivel de conocimiento.

En los tres últimos semestres de la carrera se debe de cursar Análisis y Diseño de Sistemas 2, además del curso de Software Avanzado y el curso de Seminario de Sistemas 2, de los cuales según la muestra se realizaron 71 encuestas.

La encuesta realizada contiene preguntas relacionadas al conocimiento y aplicación por parte de los estudiantes de una metodología especifica; estas preguntas buscan detectar tanto el conocimiento, interés y la importancia que tiene para los estudiantes, los temas del uso de metodologías y buenas prácticas para el PDS. Es muy importante para esta investigación conocer la opinión de los estudiantes con respecto a los 2 temas centrales de este trabajo de investigación, ya que de esa manera se hace posible el planteamiento de un proceso, en el cual los estudiantes se puedan apoyar para la gestión, administración y aplicación de un proyecto de desarrollo software cuidando aspectos de calidad y seguridad, tanto en el proceso como en los entregables del proyecto.

Con base en los resultados arrojados en la encuesta, es posible determinar que los estudiantes están conscientes de la importancia de aplicar buenas prácticas en el desarrollo de software, como lo muestran en la pregunta 8 con un notable 81,7 por ciento que opina que es muy importante aplicar de manera integral buenas prácticas en el desarrollo de software. Es posible verlo también en la pregunta 1, un 42,5 por ciento opina que tiene interés en el tema de buenas prácticas de desarrollo de software.

Los resultados muestran información clave para esta investigación, como lo es muy poco el porcentaje de los estudiantes que incluye la realización de pruebas dentro del proceso de desarrollo de software.

En la pregunta 5 en 4,7 por ciento de los encuestados afirmo que realiza pruebas en su proceso de desarrollo, asimismo admitieron un 16 por ciento de los encuestados que realizan una planificación de actividades y utilizan un sistema de versionamiento, luego un 15 por ciento que utiliza una metodología, por lo general estas son las prácticas con menor incidencia dentro del proceso de desarrollo de software de los alumnos de la carrera, y las demás prácticas todas con un porcentaje pobre, indican que realizan un diseño o documentación del proyecto.

Con base en toda la información recabada de las impresiones de los alumnos y para poder visualizar mejor los problemas y fortalezas encontrados en el PDS, se presenta en una tabla resumen las etapas definidas del proceso, en la primera columna de la tabla, seguido por las fortalezas y debilidades según la opinión de los propios estudiantes de la carrera; colocando las debilidades en negrita, con el fin de mostrar de forma inclusiva como se complementarían estas actividades dentro de cada etapa y como podrían estas debilidades en un futuro convertirse en fortalezas para el proceso de desarrollo de software.

Se hace la observación que estas etapas definidas para el PDS no son estáticas, por el contrario se consideran iterativas; asimismo las debilidades son definidas bajo el punto de vista de los alumnos encuestados y entrevistados durante este trabajo de investigación, esto no quiere decir que estas debilidades no sean parte en algunos casos del PDS de algunos estudiantes, se proponen como actividades que deben de fortalecerse para la mejora del proceso.

Tabla X. **Debilidades y fortalezas del proceso de desarrollo de software según los estudiantes de la carrera**

Toma de requerimientos	•No se conoce bien los requerimientos hasta ya iniciado el desarollo •No se priorizan los requerimientos •**Modelado de datos** •**Aprendizaje autodidacta**
Análisis y diseño	•No existe una investigación previa para para evaluar las tecnologias a utilizar •No se utiliza una ténica u orden para realizar el diseño •**Se aprende el uso de patrones de diseño como una técnica de diseño.** •**Se aprende el modelado de bases de datos tanto relacionales, transaccionales, y Data warehouse**
construcción del software	•No se tiene un control del avance del proyecto •No se tiene un control de la dimension del proyecto ni de la distribución del trabajo entre los miembros del equipo. •**El desarrollo es rapido** •**La capacidad de desarrollar sistemas es independiente de tecnologias, lenguajes de programación y arquitecturas.**
Pruebas del software	•Las pruebas no son automatizadas. •no existe un diseño de pruebas
Integración del software	•No existe una planificación de integración •No existe una planificación ni versionamiento de releases
Mantenimiento del software	•El mantenimiento no está contemplado dentro del proceso. •No existe un plan de control de cambios

Fuente: elaboración propia.

3.2.4. Entrevistas a catedráticos del área de desarrollo de software e ingenieros con experiencia en el tema del PDS

Para poder tener una idea de un PDS adecuado para los estudiantes de la carrera, se pidió la opinión a catedráticos del área de Desarrollo de Software y a Ingenieros en Ciencias y Sistemas con experiencia en el PDS, quienes compartieron sus estrategias para mitigar los problemas comunes que se presentan durante el PDS; para ello se ha creado la siguiente tabla resumen:

Tabla XI. **Problemas identificados en el PDS según la entrevista a catedráticos**

Pregunta	Elementos importantes
¿Cuál es su opinión acerca del proceso de desarrollo que se aprende en la carrera de Ingeniería en Ciencias y Sistemas?	-Problemas de coordinación entre contenidos de los cursos y los laboratorios -No existe un PDS definido, más bien es circunstancial y depende mucho de los estudiantes y su experiencia -Hace falta una guía y una recomendación para que este se realice una manera más técnica y estratégica en lugar de ser un desarrollo desordenado.
¿Con bases en su experiencia, qué recomendaciones daría usted para mejorar el proceso de desarrollo actual que siguen los alumnos de la Facultad de Ingeniería?	-Establecer uno o dos PDS, para que los estudiantes de la carrera puedan tener una opción comprobada a la hora de desarrollar un proyecto. -Definir las actividades principales dentro del PDS necesarias y adecuadas para los estudiantes
¿Qué problemas ha enfrentado y como dio solución a los mismos en el proceso de desarrollo de proyectos de software en su experiencia laboral?	-Problema en la gestión de proyectos -Problemas con la calidad del software -Problemas con la planificación -Problemas con el diseño de proyectos grandes (la presentación y documentación)

Fuente: elaboración propia.

De las entrevistas con los profesionales del área con experiencia en el PDS, surgieron puntos que pueden considerarse como estrategias ante ciertos problemas específicos y se muestran a continuación por medio de la siguiente tabla:

Tabla XII. **Problemas *versus* estrategias en el PDS**

Problema	Estrategia
El producto final no cumple con los requerimientos tomados al principio	Priorizar los requerimientos del sistema, desarrollo iterativo
El programa tiene fallas	Realizar pruebas automatizadas, utilizar integración continua, prácticas de diseño
No se estimarón bien los recursos para la realización del proyecto	Planificación, técnicas de estimación
El software no está a tiempo	Realizar un control de avances del proyecto de manera automática, seguimiento del proyecto, desarrollo iterativo
No se documentó el proyecto	Realizar la documentación poco a poco dejando hasta el final la edición de los documentos, BDD

Fuente: elaboración propia.

3.3. Problemas en el proceso de desarrollo de software

Según distintas fuentes, las críticas más fuertes hechas a la ingeniería de software giran en torno a la calidad, el control, la estandarización, la estimación de costos, tiempo y recursos entre otros. Estas críticas también se identifican con los problemas que enfrentan en el desarrollo de software los estudiantes de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala, lo cual indica que son problemas generalizados en muchos de los casos y que se puede tomar la experiencia de muchos de estos profesionales a

la hora de solucionar dichos problemas, tomar su experiencia para mejorar las debilidades dentro del proceso de desarrollo de los estudiantes de la carrera.

Estos problemas se describen de alguna manera con base en la causa de los mismos, para darle el enfoque positivo que todos los ingenieros buscan darle a los problemas, que no son más que retos, para la mejora continua;

- La mala gestión de los requerimientos; que es considerada una de las debilidades de la mayoría de los desarrolladores de software, sin embargo esta debilidad no es solo de la ingeniería de software, ya que la gestión de las expectativas del cliente es un problema o un reto al cual todos los proyectos se deben de enfrentar y satisfacer de la mejor manera posible.

    - Causas: en muchos de los casos, esto no sea más que la inexperiencia de la persona quien les transmite los requerimientos del sistema y en otros casos la malinterpretación de los requisitos de parte de los desarrolladores del proyecto (los estudiantes). Este problema también está asociado a la falta de técnica y orden en la especificación de requerimientos, es decir no existe un proceso establecido en que se le dé énfasis a la toma de requerimientos.

- Fallos constantes en los productos finales; constantemente las personas que desarrollan software de forma continua, incluyen nuevas prácticas y técnicas para aplicar en los entornos necesarios. Se critica que este desarrollo constante de nuevas prácticas es la evidencia de los fallos en las anteriores versiones.

La crítica es debida a los constantes fallos que ocurren en los sistemas con diseños o desarrollos pobres.

- Causas: en relación a esto puede argumentarse que en toda la ingeniería existen fallos, los cuales se van superando conforme maduran los productos y se va adquiriendo el conocimiento a lo largo de la carrera, el ejemplo más claro de ello es la ingeniería automotriz, en la cual existen casos en los que líneas enteras de automóviles son retirados del mercado por presentar alguna falla, lo cual no es una razón para cometer errores pero si la razón para considerar que las causas de fallos en los sistemas de software no solo de los estudiantes de la carrera, si no del desarrollo de software, en Guatemala; son la falta de control de calidad y altos estándares de calidad, mala planificación, mal seguimiento, carencia de pruebas y la no utilización de herramientas que ayuden a la automatización de pruebas y la integración continua del proyecto.

- Falta de estandarización del software; cabe mencionar esta gran crítica que gira en torno a la ingeniería de software y consiste en que en las demás ingenierías cuando sucede un error, se busca la causa y se implementa un sistema para detectarlo y que este no se vuelva a presentar; este problema es más difícil de manejar en la ingeniería de software, ya que evitar que un problema o un error se vuelva a repetir, depende de una serie de factores como por ejemplo el contexto del sistema, las variables del sistema, los módulos, tipo de sistema, plataforma, arquitectura.

Esto no significa que sea imposible, pero la mayoría de sistemas de software son muy distintos unos de otros, cada uno con distintas

funciones y la solución puede variar dependiendo de los desarrolladores del proyecto, por lo que es difícil y casi imposible, crear un sistema que detecte cada error que se encuentra; y usarlo para prevenir errores en otros sistemas; se podría incluso decir que por las características de los sistemas de software, cada uno resulta siendo casi único, puede tener un grado de parecido pero esto no significa que funcione igual e invertir en sistemas para prevenir errores en cada pequeño desarrollo implica un costo mayor.

  - Causas: frecuentemente la causa principal de este problema, es la falta de pruebas automáticas y la mala especificación de requerimientos

- La mala estimación de recursos; este es un problema generalizado en muchas áreas de la realización de proyectos, no solo en la ingeniería de software, lo cual tampoco justica la constante ocurrencia de este problema. Sucede muchas veces que el software se presenta fuera del plazo establecido, en la mayoría de los casos se entrega incompleto, y hace falta mucho tiempo para para poder terminarlo.

  - Causas**:** las causas asociadas a este problema son la mala planificación o la ausencia de la misma, la falta de conocimiento en la estimación de recursos y tiempo y la mala gestión de un proyecto.

- La modificación del software es un reto al que los estudiantes no están preparados para enfrentarse; sucede que la parte del seguimiento de un proyecto no está contemplada dentro de los conocimientos que un estudiante de la carrera de Ingeniería en Ciencias y Sistemas, adquiere

a lo largo de la carrera. Es decir no existe contenido alguno relacionado al mantenimiento de un sistema, al control de cambios más que como documento que debe incluirse como parte de un proyecto (documento de control de cambios).

- Causas: la causa de este problema como se ha mencionado, ya es la falta de atención y contenido dentro de la carrera para preparar a los estudiantes en este aspecto, tanto la gestión de cambios, el mantenimiento del software y un buen diseño .

- Integración del software y la movilidad del sistema de software: este problema más que crítica generalizada es una mala práctica que se tiene por parte de los estudiantes de la carrera y consiste en que los sistemas son difíciles de integrar, ya que existe un individualismo en el desarrollo que desemboca en pequeños subsistemas no homogéneos y difíciles de integrar. Sin mencionar la falta de una herramienta o configuración de un servidor de integración continua que favorezca esta práctica. Frecuentemente un sistema desarrollado en una máquina específica es muy difícil de trasladar a otra, sucede que el sistema que funciona en una máquina cuando es trasladado a otra máquina, ya no funciona.

  - Causas: las causas relacionadas a este problema son un poco complejas; la poca coordinación entre grupos, la individualidad de los estudiantes, resultan ser un factor muy importante a la hora de trabajar un proyecto de software en conjunto. Asimismo la falta de prácticas como la utilización de herramientas para el control de la configuración y la integración continua, se convierten en un factor clave que se evidencia en los resultados en un proyecto de software.

Es importante mencionar que también juega un factor importante el hecho de no contar con un diseño del software o tener uno muy pobre, ya que esto no permite la estandarización del desarrollo y por ende el acoplamiento de los módulos del proyecto de software, a la hora de integrarlos o modificarlos, existen varios tipos de análisis, y técnicas de desarrollo de software que lo hacen no homogéneo y muchas veces incompatible entre los módulos, lo que lo hace difícil de integrar.

Existen otros problemas que se presentan en el proceso de desarrollo muy importantes de considerar y que afectan directamente a la calidad del software como producto final del proyecto, los cuales quizás no van ligados directamente con el contenido de los cursos del área de desarrollo de sistemas de software, pero que deben de ser parte del aprendizaje o quizás como habilidades que debe adquirir un estudiante durante el transcurso de la carrera.

- La falta de comunicación. Esta puede afectar el proyecto de tal manera que este no llegue a su fin o bien no llegue a culminar de la mejor manera. Los factores que deben cuidarse mucho y en los cuales la falta de comunicación puede tener un impacto para el proyecto, se definirá como cliente en este caso, al catedrático o en su defecto auxiliar de cátedra; la situaciones más comunes en las que se da un problema de comunicación son los siguientes:

    - Las necesidades: lo que el cliente quiere, y lo que realmente necesita.

- Cliente o dueño del proyecto: lo que el cliente describe según sus conceptos y su capacidad de describir como una necesidad concreta.
- El estudiante: lo que el estudiante promete hacer dentro del grupo de trabajo.
- Los requerimientos: los requisitos o requerimientos descritos por el dueño del proyecto o cliente, tal como fueron entendidos.
- El análisis: la especificación formal de los requerimientos, según el criterio de los analistas.
- El diseño: la descripción y especificación del funcionamiento de la solución, el qué y cómo de la solución o sistema para satisfacer los requisitos analizados.
- En la codificación: lo que codifica el programador en función de lo que entendió en el diseño.
- En la instalación: lo que se instala realmente en el entorno del cliente.
- En las pruebas: las pruebas que los responsables hicieron y los resultados que obtuvieron en el sistema.

    - Causas: la causa asociada a este problema ha sido ya mencionada en algunos de los problemas del proceso de desarrollo de software, y es la coordinación de integrantes como parte de la comunicación, asimismo el carácter individual e introvertido de muchos de los estudiantes de la carrera, lo cual interfiere en la integración a grupos de trabajo.

- La falta de planificación representa un problema con gran impacto en los proyectos de desarrollo de software, ya que de los proyectos que se

realizan, tan solo un 10 por ciento son terminados completamente en el tiempo estipulado, además es muy común que se tenga que solicitar una prórroga en el tiempo de entrega, lo cual repercute en el tiempo que duran los semestres de manera oficial en la Facultad de Ingeniería. Son conocidos casos en los que se programan entregas de proyectos finales fuera del plazo del semestre.

Los casos en los que la falta de planificación tiene un impacto negativo en el proceso de desarrollo de un proyecto de software, se puede visualizar por medio de los siguientes escenarios:

- La necesidad: los estudiantes necesitan desarrollar el software en un corto tiempo para ganar un curso específico.
- Los requisitos: no se revisa los requerimientos y por ende no existe una priorización, y al final faltan requerimientos claves para el sistema.
- El estudiante: no planifica su tiempo en función de las cosas que necesita realizar y la carga de otros cursos, y lo más común es que estas se acumulen para el último momento.
- La calidad: por la misma falta de planificación, no existe calidad en el proceso y tampoco en el producto, ya que no existe un orden para la realización del software y sus funcionalidades, mucho menos un análisis de sus requerimientos y la mejor forma de realizarlos.
- El análisis: el análisis no está dentro de las actividades del proceso, por lo mismo no se puede contar con la solución más óptima al problema. Si en algún caso existe una tecnología más adecuada, esta no es considerada ya que no se ha tomado el

tiempo para investigar las opciones que se tienen respecto a tecnología. Y así en muchos casos es similar.

- El diseño: no se toma el tiempo para un diseño, por lo que el producto final no es estándar, y no tiene características de calidad que lo respalden, el tema principal es la solución del problema, no la calidad de la solución.

    - Causas: asociadas a este problema existen 2 causas principales; la primera y quizás la más importante, sea la cultura local de dejar todo a última hora. Y la segunda causa asociada a este problema, directamente enfocada a los estudiantes de la carrera, es la mala planificación y administración del proyecto, no solo a nivel del desarrollo de software, si no a planificación en general.

- La falta de automatización, durante el proceso se nota la ausencia de técnica en la solución, es muy común ver que se realizan actividades con cierto grado de automatización, pero no completamente sabiendo que a veces un proceso automático ahorra tiempo y recursos, dichas actividades se realizan en parte manualmente y la otra parte automática.

    - Causas: relacionado a este problema, está la falta de confianza en invertir tiempo en la automatización de los procesos; es el caso de la gestión del proyecto, la realización de las pruebas, el proceso de versionamiento, integración del proyecto.

Tabla XIII. **Problemas del PDS *versus* las causas de estos problemas**

	Problema	Causas
1	La mala gestión de los requerimientos	inexperiencia y malinterpretación al recibir los requerimientos, falta de técnica y orden en la toma de requerimientos
2	Fallos en los productos finales	La falta de control de calidad, mala planificación, mal seguimiento
3	Falta de estandarización de software	La falta de pruebas automáticas
4	La mala estimación de recursos	Mala planificación falta de conocimientos en estimación de recursos
5	La modificación del software es un reto al que no están preparados para enfrentar	Falta de conocimientos en el mantenimiento de un sistema, control de cambios y falta de un buen diseño
6	La falta de integración del software y la movilidad del sistema	La poca integración entre grupos , la falta de buenas prácticas, la falta de un diseño realista o un diseño de software pobre
7	Falta de comunicación	Mala coordinación de estudiantes por mala comunicación, carácter individual e introvertido por lo que se da una mala integración del equipo de trabajo
8	Falta de planificación	La cultura local de dejar todo a última hora y la mala planificación o falta de costumbre de planificar y administrar proyectos
9	La falta de automatización	Falta de confianza en el método de automatización, por desconocimiento o por falta de implementación

Fuente: elaboración propia.

3.4. Comparación del PDS de la carrera de Ingeniería en Ciencias con diferentes metodologías de desarrollo de software consideradas dentro del transcurso de la carrera

Dentro de las metodologías más conocidas por los estudiantes de la carrera se encuentran XP y RUP, ambas requieren de disciplina en su

utilización. Se hace una comparación de las actividades propuestas por estas metodologías y el PDS actual, para lo cual se propone la siguiente tabla:

Tabla XIV. **PDS *versus* metodologías de desarrollo de software**

	PDS USAC	RUP	XP
FASES	• Concepción • Elaboración • Construcción • Transición	• Concepción • Elaboración • Construcción • Transición (desarrollo en cascada)	• Planificación • Diseño • Desarrollo • Pruebas (desarrollo incremental)
VENTAJAS		• Evaluación en cada fase que permite cambios de objetivos • Funciona bien en proyectos de innovación. • Es sencillo, ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software. • Seguimiento detallado en cada una de las fases.	• Apropiado para entornos volátiles • Estar preparados para el cambio, significa reducir su coste. • Planificación más transparente para nuestros clientes, conocen las fechas de entrega de funcionalidades. Vital para su negocio • Permitirá definir en cada iteración cuales son los objetivos de la siguiente • Permite tener realimentación de los usuarios muy útil.
DESVENTAJAS		• Evaluación de riesgos compleja • Excesiva flexibilidad para algunos proyectos • Estamos poniendo a nuestro cliente en una situación que puede ser muy incómoda para él. • Nuestro cliente deberá ser capaz de describir y entender a un gran nivel de detalle para poder acordar un alcance del proyecto	• Delimitar el alcance del proyecto con nuestro cliente

Fuente: recopilación de Internet, es.scribd.com/doc/168670984 y danielzs75.blogspot.com. Consulta: 1 de mayo de 2013

3.5. Fortalezas y debilidades del PDS de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala

De acuerdo a la diferente información que se ha presentado en este trabajo de investigación, es necesario hacer un resumen uniendo las diferentes

opiniones tanto de catedráticos como de alumnos y egresados de la carrera, para lo cual presentamos la siguiente tabla resumen:

Tabla XV. **Fortalezas *versus* debilidades del PDS**

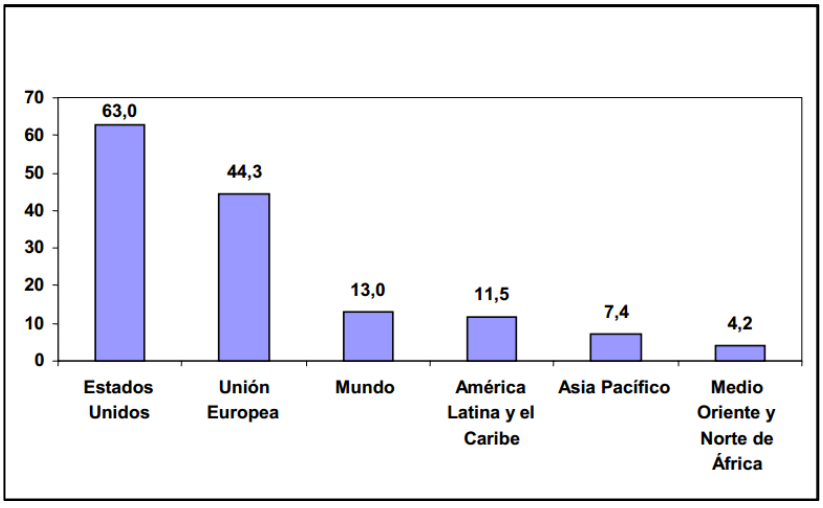
Enfoque	-POO - manejo de distintos lenguajes de programación
Fortalezas	- Aprendizaje autodidacta de los estudiantes de la carrera - Capacidad de trabajo bajo presión -Desarrollo de software independiente de la arquitectura, plataforma, lenguaje de programación, y complejidad del sistema.
Debilidades	-Administración de proyectos - Baja calidad de software -Falta de utilización de herramientas para la automatización de pruebas e integración continua, asimismo la falta de administración de la configuración -Mala estimación de recursos y tiempo

Fuente: elaboración propia.

4. PROCESO DE DESARROLLO DE SOFTWARE INCLUYENDO BUENAS PRÁCTICAS DE DESARROLLO

La ingeniería de software es una de las áreas más grandes y de más influencia en la sociedad moderna. Esta ha evolucionado desde las tempranas aplicaciones de cálculo, usadas solo por el gobierno y universidades hasta las aplicaciones complejas que permiten mantener la sociedad moderna. El software afecta desde la compra de artículos para el hogar hasta el lanzamiento de naves que llegan al espacio, según estadísticas de la EUROSTAT, el uso de la computadora es cada vez más frecuente y en un constante crecimiento; lo cual hace necesaria la construcción de software; ya sea para un uso específico o de propósito general.

Figura 41. **Usuarios de internet en regiones del mundo 2004 (en porcentajes)**



Fuente: devdata.worldbank.org/dataonline. Consulta: 1 de mayo de 2013.

Siendo la ingeniería de software una disciplina relativamente nueva, las principales críticas a la misma han sido en relación a la mala calidad de los productos que producen, esto significa que en muchos de los casos los clientes no están del todo satisfechos con los resultados; y es que la complejidad de los sistemas y las expectativas siempre van en aumento. Este problema ha sido atribuido a muchas causas, desde la mala forma en que han sido educados los profesionales de software, la forma en la que resuelven los problemas hasta culpar a los problemas propios de la ingeniería que han sido heredados dentro de la profesión.

En algunos sitios conocidos como Wikipedia, han criticado la ingeniería de software, haciendo referencia a su deficiencia en términos de control, estandarización y calidad:

"En la ingeniería tradicional, hay claras reglas de como las cosas se tienen que construir, que estándares deberían seguir los ingenieros y que riesgos deberían ser tomados en cuenta. No hay consensos en la ingeniería de software, cada ingeniero promueve sus propios métodos, reclaman grandes beneficios por productividad, usualmente no respaldados por ninguna evidencia científica. Y generalmente esas inversiones no retornan debido a estos trabajos."[1]

Asimismo, grandes desarrolladores de software, como es el caso del señor Robert C. Martin, creador de los sistemas XUNIT, o los mismos creadores de lenguajes de programación como el señor Bjarne Stroustrup, inventor de C++, y otros muchos reconocidos personajes en el mundo del desarrollo de software, han opinado acerca de la importancia de un desarrollo de calidad.

[1] El PDS, http://es.wikipedia.org/wiki/Proceso_para_el_desarrollo_de_software. Consulta: 1 de Mayo de 2013

Usando la analogía de un médico y un paciente, estos personajes son expertos en desarrollo y son reconocidos por sus grandes logros en materia de desarrollo de software; en el caso del señor Stroustrup, quien mejor que él para decir lo importante que es poner especial énfasis en el procesos de desarrollo, para que el producto final sea de gran calidad.

Estos expertos describen de alguna forma el futuro de los sistemas de software y consideran que el desarrollo o mejor dicho el código no va a desaparecer, es evidente la evolución que la tecnología, la informática y los sistemas experimentan, pero este no va a desaparecer, con su evolución lo que se debe de lograr es minimizar esas debilidades que existen durante el proceso para garantizar productos de mayor calidad.

La industria del software al igual que muchas otras crecerá cada día más y perfeccionará la técnica de su producción.

Lo anteriormente expuesto es una razón muy fuerte para querer mejorar el proceso de desarrollo que se imparte y practica durante los años de la carrera de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala; esto no es desconocido por los estudiantes de la carrera, quienes consideran importante y necesario mejorar la calidad del desarrollo, implementar buenas prácticas de desarrollo de software de manera integral.

Asimismo, lo expresan los recién egresados de la carrera afirmando que es necesario mejorar el desarrollo de software y la administración de proyectos; si bien es cierto que estos mismos consideran que el desarrollo de software es uno de los fuertes de la carrera, por lo mismo no puede descuidarse y dejarse sin la mejora continua que debe de tener todo aquello a lo que se le da valor.

Como punto principal en la mejora continua del proceso de desarrollo es necesario establecer las buenas prácticas que lleven a obtener mejores resultados y minimizar hasta eliminar esos problemas que los estudiantes de la carrera enfrentan cuando desarrollan un sistema de software.

4.1. El proceso de desarrollo de software

En general el proceso de desarrollo de software consiste en una serie de actividades comunes, las cuales deben de reforzarse con buenas prácticas para lograr los objetivos de calidad y mejora; independientemente de cómo haya sido diseñado el desarrollo de software, las fases que agrupan los subprocesos que lo componen, están directamente relacionadas a los siguientes aspectos:

- Análisis
- Especificación de requerimientos
- Diseño
- Programación
- Pruebas
- Documentación
- Mantenimiento
- Reingeniería

Asimismo, el desarrollo de software tiene diversos ciclos de vida o enfoques de desarrollo, los modelos más comúnmente utilizados son los siguientes:

- Modelo en cascada
- Modelo en espiral

- Modelo de prototipos
- Método en V
- RAD: Rapid Application Development, framework iterativo.

Durante el desarrollo de software practicado en la carrera, se presenta básicamente el modelo en cascada; es muy común ya que se desarrolla por etapas y da margen a que el desarrollo sea incremental.

4.2. Buenas prácticas

En esta sección es necesario definir un conjunto finito de buenas prácticas dentro del desarrollo de software; esto no quiere decir que estas sean las únicas buenas prácticas que existen para el desarrollo de software. Existen varios estándares de calidad que describen distintas buenas prácticas que deben aplicarse y practicarse durante el desarrollo de cierto software.

Según el enfoque de este trabajo de investigación, se pretende únicamente identificar un pequeño conjunto de esas buenas prácticas de desarrollo que le permitan a los estudiantes mejorar su aprendizaje en relación al desarrollo de software y las técnicas para ello, así como también poder obtener mejores resultados al final de sus proyectos.

Estas buenas prácticas son a nivel general enfocadas al desarrollo de un proyecto de software en los cursos de la carrera de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala, éstas mismas luego son asociadas a prácticas de metodologías definidas, lo cual permite a los estudiantes tener una mejor visión de sus alcances y el ambiente en el cual se aplican.

Las buenas prácticas de desarrollo de software que se describen a continuación, no en orden de prioridad ni mucho menos de aplicación, para ello más adelante se describe un proceso que incorpora algunas de estas, siguiendo un orden como propuesta de este trabajo de investigación.

- Buenas prácticas necesarias para el desarrollo de software

    - Primera práctica: entender los requisitos. Parece obvio, y debería serlo, pero es aquí donde comienza la mayoría de los problemas identificados durante el diagnóstico realizado. Es fundamental leer los requisitos completamente y repetidamente hasta tenerlo todo 100 por ciento claro y sin ambigüedades.

    - Segunda práctica: resaltar los requisitos clave o tomar nota de ellos en el mismo documento de requisitos o bien hacer un documento aparte. También es importante resaltar de manera distinta los puntos que son difíciles de entender.

    - Tercera práctica: realizar una lista de los requerimientos en orden de prioridad para poder abordarlos de manera más objetiva y poder planificar su desarrollo de manera adecuada.

    - Cuarta práctica: ser proactivo y solicitar asesoría y aclaraciones de parte del catedrático o auxiliar que este caso representan; el cliente de los estudiantes de la carrera, si se necesita cualquier aclaración sobre un requisito es necesario ser honesto al informar desde el principio si existe algún requisito que sea poco realista a criterio del estudiante.

- Quinta práctica: en la medida de lo posible, considerar la búsqueda de retroalimentación con los encargados de evaluar el proyecto, con el objetivo de conocer anticipadamente las impresiones y posibles errores, los cuales se pueden corregir de manera oportuna.

- Sexta práctica: realizar una planificación y un plan de gestión del proyecto con el cual todos los miembros del equipo conozcan las actividades y los tiempos para realizarlas, contemplar revisiones durante el desarrollo del proyecto.

- Séptima práctica: la definición de las pruebas debe de hacerse en paralelo junto con el análisis de requisitos antes de iniciar cualquier actividad de desarrollo. La visión que aporta el análisis de pruebas, entre el producto final y el proceso de desarrollo, es clave.

- Octava práctica: realizar una investigación en búsqueda de la mejor solución y la mejor tecnología y arquitectura acorde a los requerimientos establecidos.

- Novena práctica: comenzar a hacer código solo después de entender totalmente el diseño general del sistema.

- Décima práctica: realizar un diseño que cumpla con los estándares de calidad de desarrollo, el cual sea realista y sencillo.

- Décima primera práctica: esforzarse en cumplir actividades de revisiones y reuniones con los miembros del equipo donde se dé

una visión global del estado del proyecto y de cada una de las tareas que llevan a cabo las personas implicadas (esta es una de las prácticas que es recomendable apoyar con algún software que pueda dar estadísticas de rendimiento y avance sobre el proyecto).

- Décima segunda práctica: configurar un entorno de desarrollo que sea similar al entorno de producción o donde se valla a presentar el proyecto; es decir un entorno con las mismas características tanto de hardware como software; los estudiantes de la carrera deben de considerar la presentación del proyecto y las máquinas a utilizar, para que no se tenga problemas al momento de la presentación del software.

- Décima tercera práctica: realizar *unit tests* para cada funcionalidad, desde la más pequeña hasta la más grande. Es muy recomendable automatizar las pruebas, lo cual puede hacerse de forma temprana antes de empezar a codificar. Esto garantiza que cualquier cambio al software se estará probando automáticamente.

- Décima cuarta práctica: utilizar un sistema de versionamiento para que se realicen *updates* frecuentemente de los progresos realizados. Si se tiene cualquier dificultad o problema, es mejor no versionar software con errores dentro del sistema de versiones.

- Décima quinta práctica: haga una lista de cosas que pueden variar entre el entorno de desarrollo y el entorno de producción.

- Décima sexta práctica: elaborar una documentación de usuario con la mayor sencillez posible, esto es posible realizarlo de manera menos engorrosa si se va tomando *screenshots* y pegando dentro de un documento con una pequeña descripción para ser editado al final; lo mismo puede suceder con la documentación técnica, realizar un documento que se va dando forma poco a poco conforme se va desarrollando.

- Décima séptima práctica: realizar siempre pruebas de regresión, incluso cuando se cambia una parte muy pequeña del código.

- Décima octava práctica: utilizar un lenguaje estándar, de modelado y de diseño para que esto pueda ser comprendido por todos los miembros del equipo.

- Décima novena práctica: realizar pruebas de aceptación del proyecto para un mejor control de calidad, es recomendable hacerlo con el apoyo de un software que realice este subproceso de forma automática.

- Vigésima práctica: integrar todo el proceso de construcción de software por medio de la configuración del ambiente de desarrollo de manera que se lleve a cabo una integración continua del proyecto.

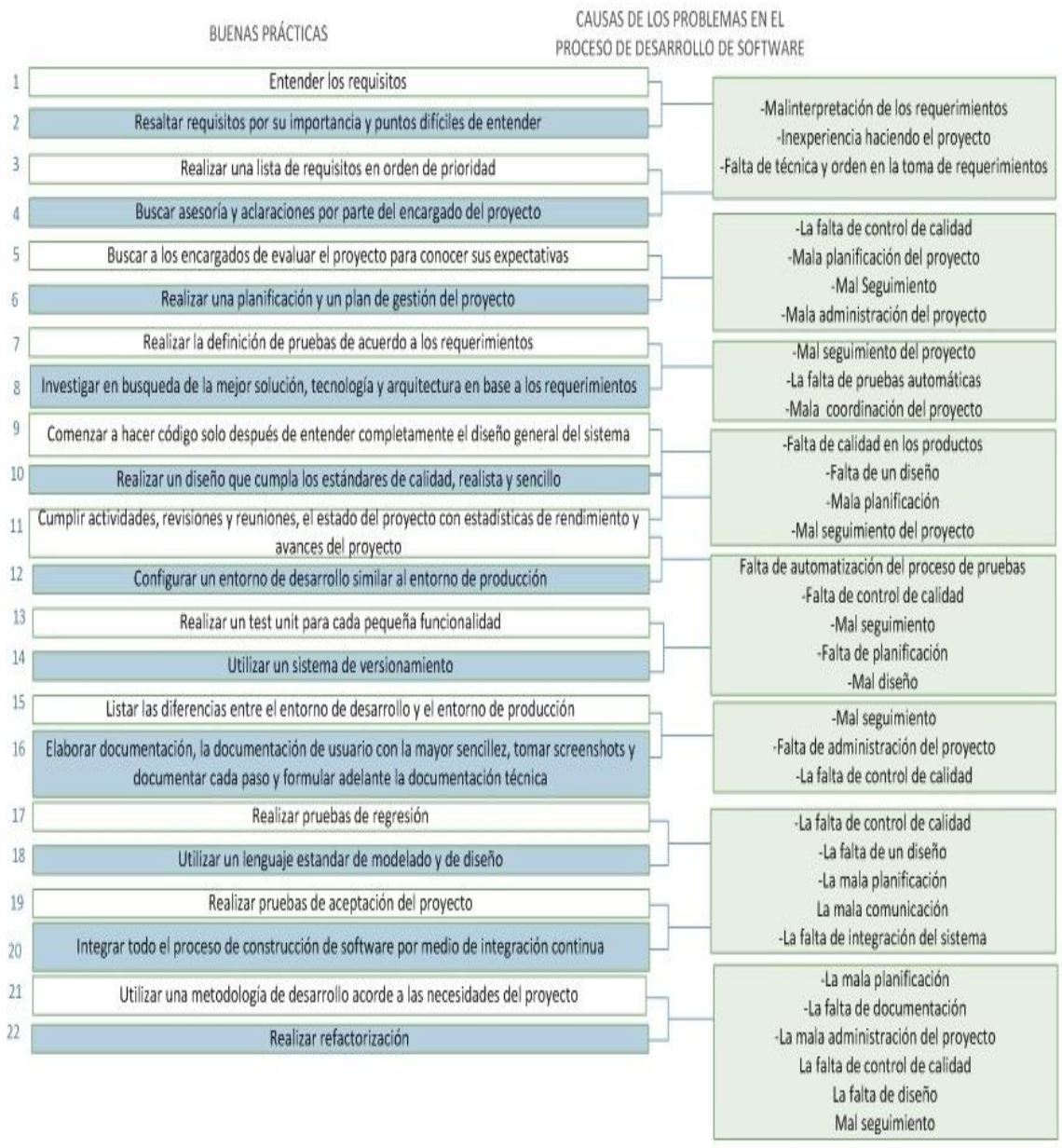
- Vigésima primera práctica: utilizar una metodología de desarrollo acorde a las necesidades del proyecto.

- Vigésima segunda práctica: refactorizar, el *refactoring* consiste en que el software debe cumplir con las expectativas de funcionalidad, pero también debe de cumplir con las expectativas de calidad. Esto se logra con la optimización de código y también de los procesos para llevar a cabo el desarrollo del producto final.

Estas buenas prácticas han sido planteadas para contrarrestar los problemas que se identificaron durante el diagnóstico realizado al proceso de desarrollo de software de la carrera; el objetivo es brindar propuestas que ayuden a mejorar el proceso de desarrollo de software y los productos finales resultado de dicho proceso; dándole la mayor cantidad de herramientas posibles para poder enfrentar los problemas que se presentan en la industria de software actual y disminuir la brecha entre los proyectos de la carrera y un proyecto real como los que se presentan en el mercado laboral.

Para observar de mejor manera como pueden estas buenas prácticas contrarrestar las causas que provocan los principales problemas dentro del proceso de desarrollo de software, se ha elaborado un mapeo de las 21 buenas prácticas propuestas en este documento y las causas que dan pie a los principales problemas que los estudiantes de la Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala, enfrentan durante el proceso de desarrollo de software, y problemas generales de esta área.

Tabla XVI. **Mapeo de buenas prácticas *versus* las causas de los problemas en el desarrollo de software**



Fuente: elaboración propia.

4.3. Metodologías de desarrollo de software

La utilización de una metodología de desarrollo de software, es muy importante si no se tiene definido un proceso de desarrollo maduro y robusto. Para los estudiantes de la carrera, las metodologías son un punto de apoyo, desde el punto de vista de la arquitectura de software y la administración de proyectos. Con estas el desarrollo toma cierto grado de técnica y madurez, lo cual se va mejorando conforme el tiempo y la práctica. Según la encuesta realizada y entrevista a los catedráticos de la carrera, las 3 metodologías más conocidas por los estudiantes son: RUP, XP y SCRUM, las cuales no son aplicadas de forma correcta cuando son consideradas dentro de un proyecto.

Una metodología de desarrollo de software o de ingeniería de software como también es conocida; es aquella guía que se sigue a fin de realizar las acciones propias de un proceso por medio de un enfoque estructurado, cuya finalidad es hacer más eficaz la producción y lograr alta calidad de acuerdo al presupuesto y recursos en general.

Como reseña histórica y análisis estructurado o Jackson Structured Design o JSD fueron de las primeras metodologías, creadas en los 70, con estas se intentaba identificar componentes funcionales básicas; ya en los 80 y 90, el desarrollo era orientado a funciones complementadas, y metodologías orientadas a objetos como las propuestas por Booch y Rumbaugh, integrando elementos de estandarización de lenguaje basados en UML.

El papel que juegan las metodologías es sin duda esencial en un proyecto y en el paso inicial, que debe encajar en el equipo, guiar y organizar actividades que conlleven a las metas trazadas en el grupo. La alta necesidad de que los proyectos lleguen al éxito y obtener un producto de gran valor, generan grandes

cambios en las metodologías adoptadas por los equipos para cumplir sus objetivos, puesto que, unas se adaptan mejor que otras, al contexto del proyecto brindando mejores ventajas. El éxito del producto depende en gran parte de la metodología escogida por el equipo, ya sea tradicional o ágil, donde los equipos maximicen su potencial, aumenten la calidad del producto con los recursos y tiempos establecidos.

Es importante mencionar que no existe una metodología perfecta e ideal, metodologías distintas tienen también distintas áreas de aplicación. Por ejemplo los métodos orientados a objetos, comúnmente son apropiados para ser aplicados para sistemas interactivos pero no para sistemas con tiempos rigurosos de tiempo real. El tema principal gira en torno a qué metodología usar, y para ello muchos expertos dan su opinión señalando que la elección entre una metodología ágil y una rígida depende directamente de los requerimientos del sistema y las condiciones del proyecto, razón por lo cual debe de abordarse ambos tipos de metodologías dentro del contenido de los cursos de la carrera.

Analizando más a fondo el problema y con la panorámica de las necesidades de los estudiantes de la carrera, cabe mencionar que lo importante no radica en que metodología utilizar, al menos no como tema central, más bien el utilizar una metodología como primer paso y de la forma correcta, es decir los estudiantes saben de la existencia de una metodología, pero muchas veces es una idea errónea de la misma, como es el caso de su opinión en la encuesta referente a la pregunta relacionada con las características de una metodología específica (XP), en la que la mayoría expresa que no se debe de documentar el proyecto. En síntesis para resolver parte de los problemas de administración y gestión de un proyecto, es la utilización de una metodología pero si esta no está bien aplicada, los resultados seguirán siendo los mismos.

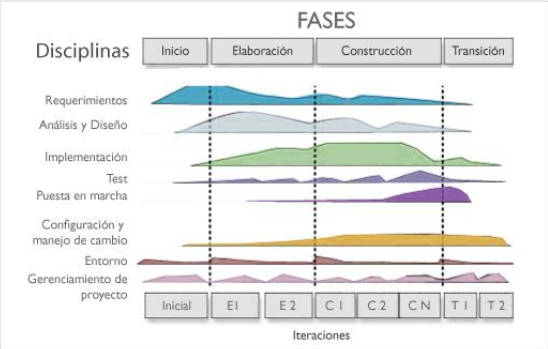
4.3.1. Desarrollo rígido o tradicional

A raíz de la necesidad de mejorar el proceso y llevar los proyectos a la meta deseada, tuvieron que importarse la concepción y fundamentos de metodologías existentes en otras áreas y adaptarlas al desarrollo de software. Esta etapa de adaptación contenía el desarrollo dividido por etapas de manera secuencial, que mejoraba la producción de software

Entre las principales metodologías tradicionales se tienen RUP, MSF, CMMI entre otros, que centran su atención en llevar una documentación exhaustiva de todo el proyecto y cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto.

Una de las metodologías rígidas más conocidas para los estudiantes de la carrera es RUP, la cual tiene como fortaleza la documentación y control del proceso de desarrollo. También es necesario mencionar que estas características no garantizan la calidad del producto. Para observar de mejor manera esta metodología se presenta la siguiente figura.

Figura 42. **Metodología RUP**



Fuente: www.epidataconsulting.com, consulta 1 de mayo de 2013.

.CMMI es un grupo de modelos desarrollados por el Software Engineering Institute (SEI), los cuales sirven como guía para alcanzar la optimización permanente en procesos de ingeniería y otros relacionados; a través de niveles de evolución o madurez. Los niveles de madurez propuestos por el modelo CMMI son:

- NIVEL 5: optimización permanente y se encuentra el estado más alto o estado final de este modelo.

- NIVEL 4: control estadístico, que se refiere a que se ha logrado un control estadístico de los procesos, de los cuales se puede conocer su estado actual y su futuro; es decir que se puede medir cuantitativamente y controlar los resultados.

- NIVEL 3: estandarización del proceso, este nivel implica que los procesos son estándares y se sigue un mismo método para su práctica, esto implica que son por la organización y la pro actividad,

- NIVEL 2: gestión de proyectos, proceso caracterizado por proyectos y administración.

- NIVEL 1: nivel inicial, proceso impredecible, pobremente controlado y reactivo.

El siguiente cuadro resumen describe algunas de las áreas de proceso relacionadas a los niveles de madurez que abarca el CMMI, que son utilizadas para el desarrollo de software.

Tabla XVII. **Áreas de proceso de CMMI relacionadas al PDS**

Área de proceso	Categoría	Nivel de Madurez
Análisis y Resolución Causales (CAR)	Soporte	5
Análisis y Resolución de Decisiones (DAR)	Soporte	3
Aseguramiento de la Calidad de Procesos y Productos (PPQA)	Soporte	2
Desarrollo de Requerimientos (RD)	Ingeniería	3
Administración de Acuerdos con Proveedores (SAM)	Ingeniería	2
Administración de Requerimientos (REQM)	Gestión de proyectos	3
Administración de Riesgos (RSKM)	Soporte	2
Administración de la Configuración (CM)	SOPORTE	2
Integración de Producto (PI)	Ingeniería	3
Medición y Análisis (MA)	Soporte	2
Monitoreo y Control de Proyecto (PMC)	Gestión de proyectos	2
Planificación de Proyecto (PP)	Gestión de proyectos	2
Solución Técnica (TS)	Ingeniería	3
Validación (VAL)	Ingeniería	3
Verificación (VER)	Ingeniería	3

Fuente: www.monografias.com/trabajos56/modelo-cmmi, Consulta: 1 de mayo de 2013.

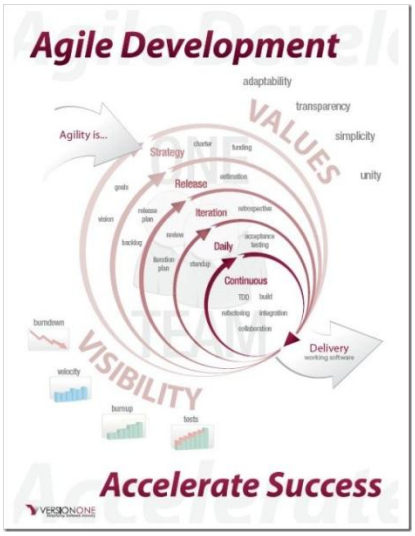
4.3.2. Desarrollo ágil

Este tipo de desarrollo es actualmente el más popular dentro de la metodología de desarrollo de software en la comunidad Java (al menos según la encuesta de lectores TheServerside.com 2011). Un total del 52 por ciento de los encuestados está utilizando actualmente desarrollo ágil y otro 20 por ciento ve un cambio de este estilo de desarrollo en el futuro.

Los métodos ágiles, nacen como respuesta a los problemas que se presentan con el desarrollo rígido, como por ejemplo, un desarrollo que toma mucho tiempo en tener un prototipo del sistema. Se basa en 2 aspectos puntuales, el retrasar las decisiones y la planificación adaptativa; permitiendo potenciar aún más el desarrollo de software a gran escala con resultados en corto tiempo.

Este tipo de desarrollo cuenta con muchas características y facetas que lo hacen tan popular dentro de la comunidad de desarrolladores, como se muestra en la siguiente figura, puntos de vista de qué es el desarrollo ágil y los valores que este aporta.

Figura 43. **Metodologías de desarrollo ágil**



Fuente: geeks.ms/blogs/rcorral/archive/2007/09/27, consulta: 5 de mayo de 2013

El valor de este enfoque está siendo probado una y otra vez con la mayoría de los usuarios, quienes coinciden que las características más valiosas

de la implementación de procesos ágiles son: los costos más bajos, ciclos de entrega más cortos y los tiempos de liberación más rápidos.

Los beneficios se extienden en virtud a una mayor flexibilidad en cuanto a la gestión de cambios y el dinamismo de un equipo que no requiere de tanta supervisión, pero fortalece la comunicación entre los interesados.

Cuando se descompone la muestra en más subcategorías, el 46 por ciento de los encuestados se inclinan por la técnica de *Scrum* que es bastante conocido, mientras que otros se inclinan por otras ramas menos conocidas del árbol de desarrollo ágil. Muchos de estos se utilizan realmente en forma híbrida, ya que no hay mucha libertad para mezclar y combinar con base en las necesidades de una determinada organización, equipo o proyecto.

4.3.3. Buenas prácticas propuestas por algunas metodologías

Las buenas prácticas es un tema importante dentro de las metodologías de desarrollo de software, ya que todas buscan la calidad del software y minimizar los riesgos de errores y contratiempos, utilizando técnicas probadas que den buenos resultados.

Como se ha dado a conocer durante este trabajo de investigación, existen metodologías de desarrollo rígidas y ágiles; dentro de estas dos categorías existen varias metodologías con un enfoque distinto, por ello debe de considerarse utilizar una metodología específica de acuerdo a las necesidades del proyecto.

Tabla XVIII. **Buenas prácticas de desarrollo de software en función de metodologías de rígidas**

RUP	CMMI
El enfoque de esta, es relacionado más con asuntos de negocio que con aspectos técnicos de la creación de software.	El enfoque de CMMI se dirige a desarrollar la madurez basada en el concepto de la evolución de los procesos dentro de las organizaciones de desarrollo de software en forma progresiva nivel a nivel.
1. Desarrolle el software de forma iterativa. Planifique incrementos del sistema basados en las prioridades del usuario y del desarrollo. Entregue las características del sistema de más alta prioridad al inicio del proceso de desarrollo. 2. Gestione los requerimientos. Documente explícitamente los requerimientos del cliente y manténgase al tanto de los cambios de estos requerimientos. Analice el impacto de los cambios en el sistema antes de aceptarlos 3. Utilice arquitecturas basadas en componentes. Estructure la arquitectura del sistema en componentes. 4. Modele el software visualmente. Utilice modelos gráficos UML para presentar vistas del software. 5. Verifique la calidad del software. Asegure que el software cumple los estándares de calidad organizacionales. 6. Controle los cambios del software. Gestione los cambios del software usando sistemas de gestión de cambios y procedimientos y herramientas de gestión de configuraciones.	Véase tabla X, Áreas de proceso de CMMI

Fuente: elaboración propia.

Tabla XIX. **Enfoque de las metodologías ágiles de desarrollo de software**

XP	*SCRUMBAN*	*PSP*
El enfoque de XP es el desarrollo rápido y que tenemos resultados o productos frecuentemente	Esta es una mezcla de dos metodologías, Scrum es una metodología que busca el trabajo colaborativo acompañado de buenas prácticas para la gestión de proyectos, se trabaja con entregas parciales y una entrega final, por otro lado kanban que es más un sistema de control de producción.	PSP facilita el desarrollo incremental. Para grandes proyectos, cada incremento puede ser un proyecto PSP entero, una fase de PSP o parte de una fase, dependiendo de las necesidades del proyecto.

Fuente: elaboración propia.

Tabla XX. **Buenas prácticas de desarrollo de software en función de metodologías de ágiles**

XP	***SCRUMBAN***	***PSP***
1. Programación en parejas 2. 40 horas semanales 3. Integración continua 4. Estándares de codificación 5. Propiedad colectiva 6. Pruebas automatizadas 7. Pequeñas liberaciones 8. El juego de la Planificación 9. Metáfora 10. Diseño simple 11. Refactorizar 12. Cliente en el lugar	• *Scrum* 1. Trabaja con Sprints 2. Planifica cada iteración 3. Se ejecuta la iteración 4. Reunión diaria con el equipo de trabajo 5. Demuestran los requisitos completados 6. Se realiza una retrospectiva 7. Se re-planifica el proyecto • *Kanban* 1. Visualizar el trabajo y las fases del ciclo de producción o flujo de trabajo. En tarjetas pegadas en una pizarra. 2. Determinar el límite de "trabajo en curso" (Work In Progress) 3. Medir el tiempo en completar una tarea	1. Planeamiento: producir un plan para hacer el trabajo. < artefactos a producir> 2. Desarrollo: Definición de requerimientos. <algunos artefactos para esto son CU, modelo de dominio, etc.> -Diseño de programa <arquitectura de todo> - Revisión del diseño y arreglar todos los defectos <revisión con colegas, maestros, etc.> - Codificación del programa -Revisión y limpieza de errores de código. - Integración y limpieza de errores de integración. - Pruebas de usuario y limpieza de errores de lógica. Post-mortem: comparar ejecución actual contra el plan, recolectar datos del proceso como mediciones, patrones, etc., producir reportes y documentar todas las ideas para mejorar el proceso.

Fuente: elaboración propia.

4.4. Proceso de desarrollo integrando buenas prácticas de desarrollo de software

Si bien es cierto no existe una receta con la cual se pueda garantizar un proceso de desarrollo de software que dé como resultado un producto 100 por ciento de calidad, es posible proponer un proceso de desarrollo en el que de manera integral se apliquen buenas prácticas de desarrollo bajo el marco de

una metodología de desarrollo de software, como lo dice el título de este trabajo de investigación.

Un proceso definido es una secuencia de pasos documentada requerida para realizar un trabajo. Los procesos definidos son usuales en trabajos que se repiten y necesitan realizarse siempre de la misma forma. Un proceso definido provee:

- Un marco de trabajo para planear, seguir y manejar trabajo.
- Una guía para realizar el trabajo correctamente y completamente.
- Una base objetiva para medir el trabajo y los procesos de acuerdo a los objetivos, y para refinar el proceso en futuras iteraciones.
- Una herramienta para manejar la calidad de los productos producidos.
- Un mecanismo para que todos los miembros de un equipo puedan soportar todas las tareas de los procesos, esto es muy importante debido a que, en la mayoría de empresas un miembro del equipo soporta todo el proceso y cuando este deja de trabajar en el proyecto, el proyecto difícilmente continúa, o difícilmente continúa con la calidad inicial, si este contaba con una calidad definida.

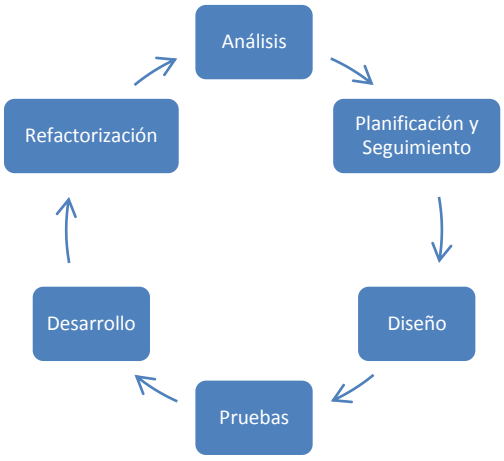
La documentación de procesos consiste en producir representación escrita de los procesos, las fases del proceso, las etapas de las fases, esta documentación no debe contener tutoriales ni manuales de usuario, solo información necesaria para comprender el proceso y el producto.

Se ha recomendado en el epígrafe 4.2 de este mismo capítulo, utilizar una serie de buenas prácticas de desarrollo, que constituyen el proceso de desarrollo de software que se presenta a continuación; no obstante la utilización de una metodología de desarrollo de software u otra, es más difícil de definir.

Como se señala en el epígrafe 4.3 de este capítulo, la elección de una metodología no es cuestión de preferencia si no de las condiciones y requerimientos de un proyecto de software y más que elegir una metodología el objetivo es aplicarla correctamente.

Como es normal en este tipo de procesos se establece el enfoque que este proceso de desarrollo debe tener, el cual es un hibrido que corresponde a un desarrollo en espiral y se combina con un desarrollo iterativo incremental, lo que significa que esta espiral se repite con distintas iteraciones hasta tener un producto final de calidad.

Figura 44. **Etapas del desarrollo de software**



Fuente: elaboración propia.

Las etapas mostradas en este modelo son descritas de manera más puntual a continuación:

- Análisis

Es la base de la construcción de un software, ya que si este no se realiza de forma correcta, las repercusiones son de gran impacto, y se pueden considerar la base de los problemas en el proceso de desarrollo de software de la carrera, pero también es clave el tiempo que este debe de llevar para desarrollar de manera ágil. Lo recomendable en esta actividad, es que se cumpla con las primeras buenas prácticas de desarrollo de software, leyendo y comprendiendo el proyecto al 100 por ciento, y solicitar todas las aclaraciones convenientes relacionadas al mismo.

Es importante construir durante esta fase un instrumento que defina el alcance del proyecto, se trata de una lista de requerimientos ordenada según la prioridad de los mismos y posteriormente realizar una investigación de tecnologías adecuadas para poder solventar cada uno de los requerimientos especificados. Para verlo de manera más clara se dice que en esta fase deben implementarse las prácticas:

- La primera práctica: entender los requisitos
- La segunda práctica: resaltar requisitos por su importancia y puntos difíciles de entender
- La tercera práctica: realizar una lista de requisitos en orden de prioridad
- La octava práctica: investigar en búsqueda de la mejor solución, tecnología y arquitectura con base en los requerimientos

Como parte de la implementación de calidad dentro de este proceso, se recomienda establecer métricas y escenarios para cada uno

de los requerimientos funcionales y no funcionales del proyecto. Se debe recordar que cada requerimiento del proyecto debe ser medible y comprobable para establecer la medida de calidad que este tenga, para ello es importante tomar en cuenta las expectativas del cliente o de parte de las personas encargadas de evaluar el proyecto, para lo cual es recomendable también incluir prácticas:

- La cuarta práctica: buscar asesoría y aclaraciones por parte del encargado del proyecto
- La quinta práctica: buscar a los encargados de evaluar el proyecto para conocer sus expectativas
- La décima primera práctica: cumplir actividades, revisiones y reuniones, el estado del proyecto con estadísticas de rendimiento y avances del proyecto

- Planificación y seguimiento

Es en esta fase es cuando la mayor parte de la planeación para el proyecto es terminada. El alumno o el equipo preparan las especificaciones funcionales, y prepara los planes de trabajo, estimaciones de recursos y cronogramas de los diferentes entregables del proyecto, cumpliendo con las siguientes prácticas:

- La quinta práctica: realizar una planificación y un plan de gestión del proyecto
- La sexta práctica: realizar la definición de pruebas de acuerdo a los requerimientos
- La vigésima práctica. integrar todo el proceso de construcción de software por medio de integración continua

- La vigésima primera práctica: utilizar una metodología de desarrollo acorde a las necesidades del proyecto

- Administración del tiempo

Para practicar la administración del tiempo, problema muy frecuente en el proceso de desarrollo de software para los estudiantes de la carrera, es necesario comprender en primer lugar como se utiliza el tiempo. Para este fin se necesita seguir los siguientes pasos:

- Categorizar la mayoría de las actividades
- Registrar el tiempo que se pasa en tales actividades.
- Registrar este tiempo en una forma estándar
- Guardar estos registros en un medio fácil de utilizar.

Seguido de esto es necesario, hacer un seguimiento a la planificación del tiempo, esto es posible hacerlo por medio de la construcción de métricas y una unidad representativa del trabajo, por ejemplo un sistema moderno de hacerlo es el análisis basado en LOC, este análisis tiene como finalidad cuantificar el trabajo realizado en un tiempo determinado y conocer el esfuerzo realizado por una persona en términos de líneas de código LOC.

Basados en este análisis, es útil mencionar que se puede estimar tendencias que ayudarán a corregir eventos que causaron un impacto el desarrollo del sistema y a su vez evaluar el rendimiento y eficiencia de cada uno de los módulos del sistema.

- Proceso de control

El proceso de control involucra desde la gestión del proyecto como tal, hasta el control mismo de calidad del producto, es decir el control del proceso, que debe de seguirse de acuerdo al alcance del proyecto y los resultados deseados. Con herramientas de apoyo para el monitoreo del avance del proyecto y los resultados de las pruebas de rendimiento y funcionalidad.

- Estimación

La capacidad de estimación es un factor determinante para el éxito del proyecto, es necesario que se puedan estimar los recursos relacionados al proyecto y el alcance del proyecto con mucha precisión y realismo, no es bueno para la reputación de los estudiantes de la carrera que no puedan establecer el tiempo o los recursos que necesitan para desarrollar un software. En esta misma sección se ha descrito una técnica para la estimación del tiempo y la planificación. También es necesario poder estimar el alcance del proyecto de forma correcta, establecer métricas que puedan evaluar al final el producto resultante del proceso.

Debe documentar de forma mínima - prueba de concepto; la descripción del contenido de la prueba de concepto, los diversos escenarios a simular, los criterios de validez, el control de incidencias y las métricas de calidad, son objetivos a cubrir en este documento. Este documento debe ser dinámico, en el que se recoge la idea y la experiencia práctica al llevarla a cabo en entorno controlado y aislado.

- Diseñar

El diseño es la parte en la que se evalúa y encuentra una solución a los requerimientos del sistema de forma viable. Se crea una estructura que organiza la lógica del sistema, un buen diseño permite que el sistema crezca con cambios en un solo lugar. Los diseños deben ser sencillos, si alguna parte del sistema es de desarrollo complejo, lo apropiado es dividirla en varias.

Se debe de recordar que el diseño debe tener un lenguaje estándar, y debe ser comprensible hasta por alguien que no conoce el proyecto. Si hay fallos en el diseño o malos diseños, estos deben ser corregidos cuanto antes. Durante esta etapa del desarrollo de software se recomienda utilizar algunas de las buenas prácticas propuestas en este trabajo de investigación, tales como:

- La séptima práctica: realizar la definición de pruebas de acuerdo a los requerimientos
- La décima práctica: realizar un diseño que cumpla los estándares de calidad, realista y sencillo
- La décima primera práctica: utilizar un sistema de versionamiento
- La décima cuarta práctica: listar las diferencias entre el entorno de desarrollo y el entorno de producción
- La décima sexta práctica: elaborar documentación; la documentación de usuario con la mayor sencillez, tomar *screenshots* y documentar cada paso y formular adelante la documentación técnica
- La décima octava práctica: utilizar un lenguaje estándar de modelado y de diseño

- La décima novena práctica: integrar todo el proceso de construcción de software por medio de integración continua
- La vigésimo primera práctica: utilizar una metodología de desarrollo acorde a las necesidades del proyecto

- Hacer pruebas

Las características del software que no pueden ser demostradas mediante pruebas simplemente no existen. Las pruebas dan la oportunidad de saber si lo implementado es lo que en realidad se tenía en mente. Las pruebas indican que el trabajo funciona, cuando no se puede pensar en ninguna prueba que pudiese originar un fallo en el sistema, entonces se habrá acabado la búsqueda.

Es recomendable diseñar las pruebas conjuntamente con el análisis de requerimientos, estas puede codificarse utilizando herramientas que ayudan a su automatización, antes de iniciar con el desarrollo, es decir; puede implementarse la técnica de *Test Driven Developement*, la cual consiste en diseñar las pruebas, desarrollarlas, para luego desarrollar el código y estas pruebas sean ejecutadas, y por último la refactorización de código. Las buenas prácticas necesarias durante esta etapa son:

- La décima tercera práctica: realizar un *test unit* para cada pequeña funcionalidad
- La décima séptima práctica: realizar pruebas de regresión
- La décima novena práctica: realizar pruebas de aceptación del proyecto

- Automatización de pruebas

Durante la fase de diseño se ha recomendado diseñar paralelamente las pruebas del sistema, estas deben ser en relación a los requerimientos funcionales, no funcionales, pruebas de aceptación. Esto es posible llevarlo a cabo con el apoyo de un software que facilite este tipo de actividad.

  - TDD

    La utilización de la metodología de *Test Driven Developement* (TDD), consiste en realizar luego del diseño de las pruebas, codificar pruebas unitarias, las primeras son pruebas de concepto del sistema, es decir las pruebas de construcción del sistema (creación, parámetros, variables, funciones.) y luego desarrollar el código para que estas pruebas sean aplicadas. Luego de ello debe de desarrollarse las pruebas verdes o de funcionalidad; estas consisten en pruebas que garanticen el adecuado comportamiento del sistema, que haga lo que deba de hacer de manera correcta. Para luego desarrollar el código que evaluarán estas pruebas.

- Desarrollo

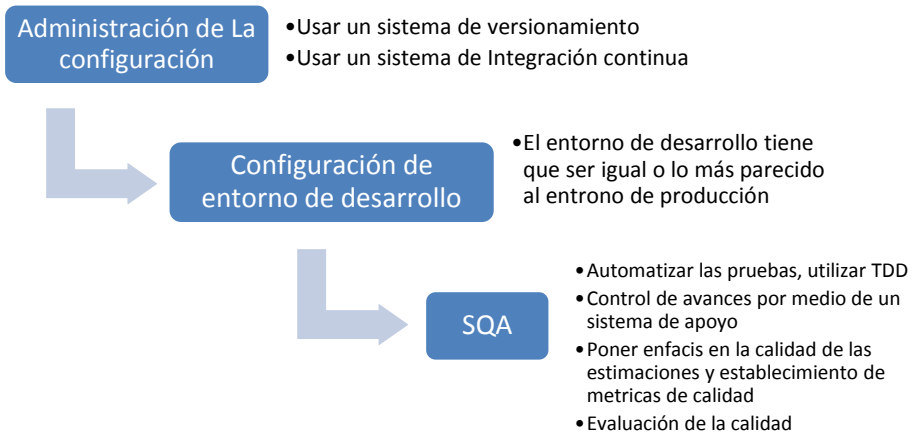
Durante esta fase, el equipo realice la mayor parte de la construcción de los componentes (tanto documentación como código), sin embargo, se puede realizar el mismo de manera incremental, por ejemplo en el caso de la documentación, debe iniciar por documentar las actividades de análisis y planeación, teniendo las especificaciones de requerimientos y de pruebas a realizarse. Al iniciar a escribir código, es

importante documentar de forma sencilla el código. Y paralelo a ello ir formando un documento en el que con *screenshots* se vaya detallando el funcionamiento, para finalizar dándole formato y edición a dicho documento. El código también puede desarrollarse de manera incremental, es decir desarrollar prototipos a los que se pueda añadir funcionalidad. Es importante mencionar las prácticas relacionadas a esta etapa:

- La novena práctica: comenzar a hacer código solo después de entender completamente el diseño general del sistema.
- La décima segunda práctica: configurar un entorno de desarrollo similar al entorno de producción.
- La décima sexta práctica: elaborar documentación, la documentación de usuario con la mayor sencillez, tomar *screenshots* y documentar cada paso y formular adelante la documentación técnica.
- La vigésima práctica: integrar todo el proceso de construcción de software por medio de integración continua.

Es importante enfatizar en la seguridad y calidad del proceso, para lo cual es recomendable tomar las siguientes medidas

Figura 45. **Medidas para asegurar la calidad del proceso de desarrollo**



Fuente: elaboración propia.

- Administración de la configuración

Es necesario tener un ambiente de desarrollo controlado, lo cual significa minimizar la posibilidad de errores. Para lo cual es recomendable utilizar herramientas que faciliten el versionamiento del proyecto, para ello existen varias de uso libre como por ejemplo GIt, configurando un repositorio central y ramas, lo cual permitirá tener un mejor control de las versiones y evitar que se pierda o sobre escriba código.

También es recomendable utilizar una herramienta de integración, lo cual facilita la liberación de software. La liberación de software en este

proceso de desarrollo debe de hacerse seguido, ya que el desarrollo es iterativo incremental y facilita que se den prototipos con cierto grado de funcionalidad antes de la liberación final.

- Configuración de un entorno de desarrollo

  El entorno de desarrollo es recomendable que sea lo más parecido al entorno de producción; esto evitará que el software falle cuando sea trasladado al entorno en el que será evaluado. Este debe definirse desde el inicio, durante la planificación, en qué máquina o máquinas funcionará el software.

- Software Quality Assurance (SQA)

  Se recomienda utilizar algunas buenas prácticas para obtener una mejor calidad del proceso y de sus resultados como producto final del mismo.

- Refactorización

  Refactorización del código, es decir; reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.

4.5. Esquema del proceso propuesto

Con base en los elementos planteados anteriormente para un proceso de desarrollo más eficiente, se propone un esquema de proceso que ayude a

contrarrestar los problemas que se dan durante el proceso de desarrollo de software actual.

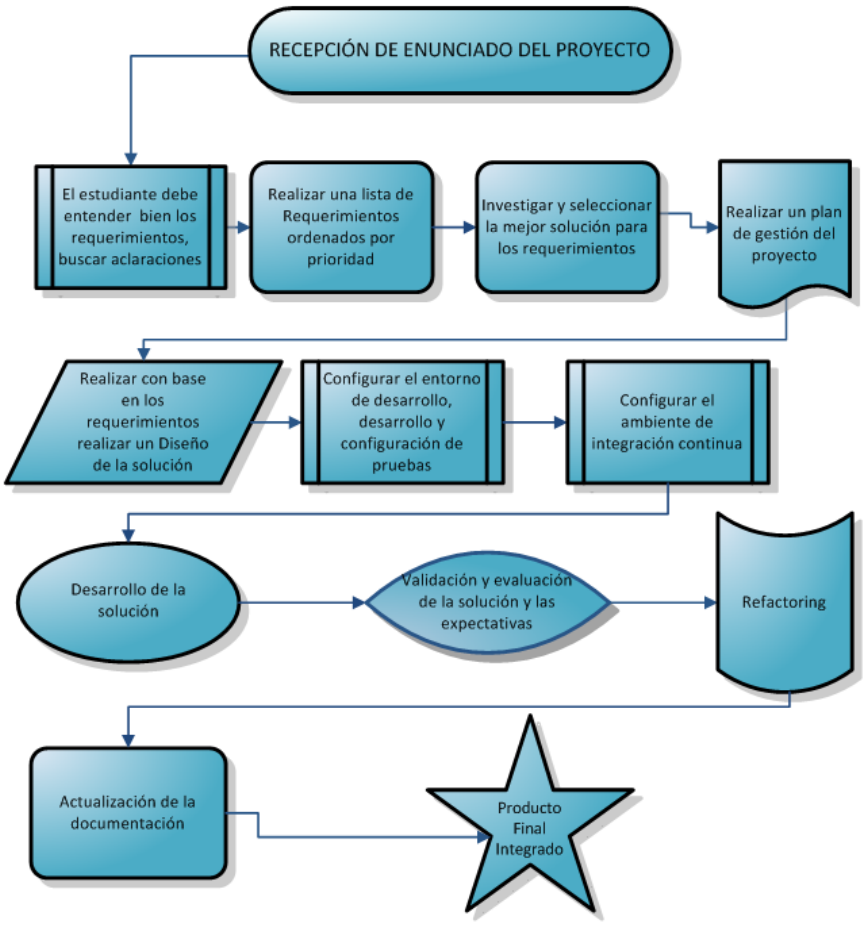
Este es un esquema básico de desarrollo, el cual es el primer paso y no debe de tomarse como un proceso estático, por el contrario la mayoría de las actividades pueden ser iterativas y es necesario que este sea fortalecido incluyendo buenas prácticas de desarrollo de software conforme al avance y el conocimiento de los estudiantes de la carrera de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala, incluyendo buenas práctica de desarrollo de software y actividades de control y administración de proyectos.

4.6. Objetos de aprendizaje para el PDS

Los objetos de aprendizaje que sustentan el proceso de desarrollo de software propuesto, están basados en reforzar las áreas de calidad, control y gestión; este ha sido propuesto basado en el actual *pensum* de estudios de la carrera, únicamente agregando algunos objetos de aprendizaje necesarios según las debilidades identificadas dentro del proceso de desarrollo de software y modificando un poco el orden en que estos son brindados a los estudiantes para una mejor comprensión y asimilación.

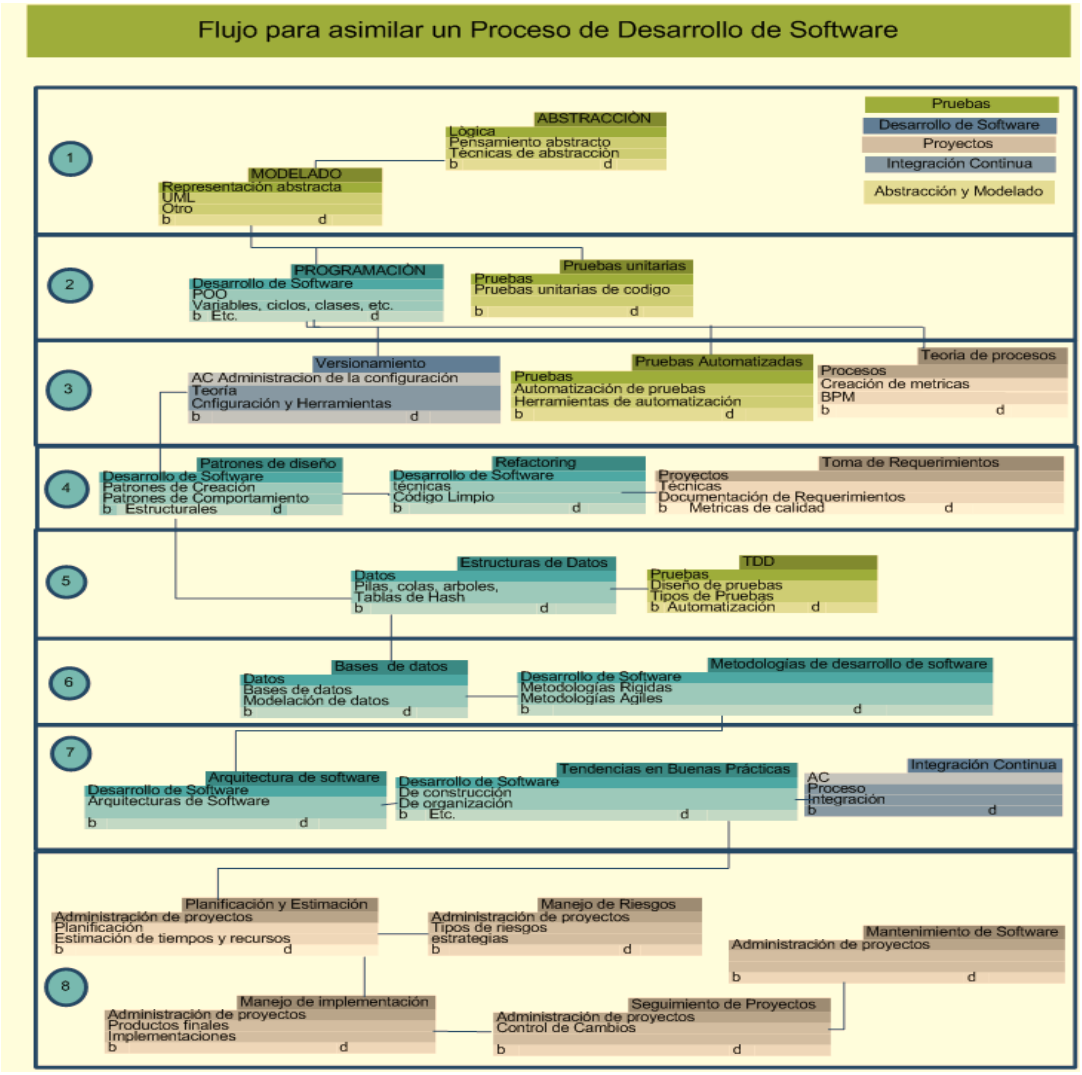
El orden de estos objetos de aprendizaje ha sido considerado por profesores del área como el indicado de acuerdo a los objetos de aprendizaje que se desean brindar a los estudiantes. Para observar mejor los objetos de aprendizaje se presenta el siguiente diagrama.

Figura 46. **Diagrama de flujo básico del PDS propuesto**



Fuente: elaboración propia.

Figura 47. **Objetos de aprendizaje para asimilar mejor un PDS**

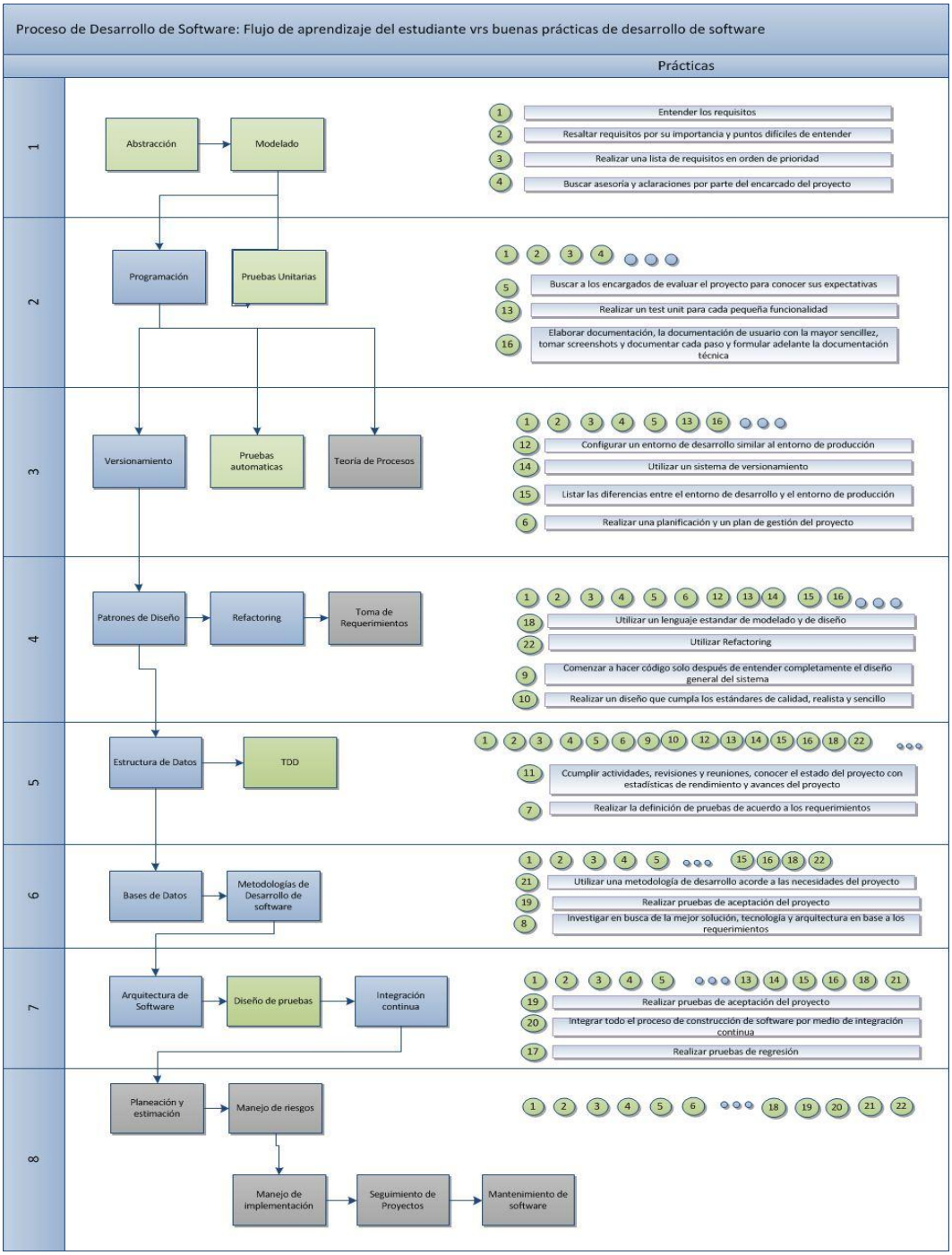


Fuente: elaboración propia.

4.7. Objetos de aprendizaje para el PDS incluyendo buenas prácticas de desarrollo de software

Esta es una propuesta para integrar buenas prácticas dentro de los objetos de aprendizaje del proceso de desarrollo de software.

Figura 48. **Objetos de aprendizaje para el PDS incluyendo buenas prácticas de desarrollo de software**



Fuente: elaboración propia.

5. ESTRATEGIAS PARA MEJORAR EL PDS

Con base en el objetivo de apoyar a la mejora y actualización del proceso de formación de los ingenieros en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala, específicamente en el área de desarrollo de software, y habiendo identificado las fortalezas y debilidades del proceso de desarrollo de software, se ha propuesto en el capítulo anterior objetos de aprendizaje y buenas prácticas de desarrollo que es necesario implementar. Asimismo, se sugiere un seguimiento y evaluación para esta inserción dentro del proceso de formación de los estudiantes de la carrera que refleje a mediano plazo la eficiencia del proceso que se utiliza y los posibles puntos en los que se deba mejorar.

5.1. Evaluación y seguimiento del cambio

Para evaluar si el proceso de desarrollo y las buenas prácticas ayudan a mejorar la calidad, tanto del proceso de desarrollo de software como de los productos finales de dichos procesos, es necesario establecer elementos de control y monitoreo, con los cuales se obtenga una panorámica de la evolución del proceso de desarrollo y de la implementación de las buenas prácticas recomendadas. Los instrumentos de control y monitores son los siguientes:

- Una hoja de evaluación de progreso; en la que se evalué los resultados del proceso, y el proceso de desarrollo de software como tal. Esta debe de aplicarse según el diseño de muestra a los cursos finales del área de Desarrollo de Software (Análisis y Diseño de Sistemas 1 y 2, Seminario de Sistemas 2, y Software avanzado). Dicha hoja puede servir de

retroalimentación para los catedráticos de los cursos como a la escuela para evaluar el rendimiento inter ciclo y la evolución de acuerdo a cambios que se implementen en el contenido de los cursos.

- La segunda es una gráfica de control; la cual muestra de forma gráfica un criterio en que se evalúa por ciclo académico si los resultados están o no dentro del rango de aceptación, estableciendo un rango mínimo y un máximo para dicha evaluación.

5.1.1. Hoja de evaluación

La hoja de evaluación evalúa los siguientes puntos críticos dentro del proceso de desarrollo de software, para los cuales debe de evaluarse de 1 a 10 el valor promedio con base en los resultados de los estudiantes en los siguientes criterios:

- La calidad del proceso de desarrollo.
- Se completan los productos en tiempo estimado.
- Se cumplen los requerimientos estipulados al inicio del proyecto.
- Las estimaciones teóricas concuerdan con la evaluación en tiempo real.
- Se tiene la menor incidencia de fallos en los sistemas presentados.
- Se tiene la menor incidencia de fallos en el proceso de desarrollo.
- Se utiliza correctamente una metodología de desarrollo.
- Se implementan buenas prácticas de desarrollo de software.
- Se utiliza un sistema automatizado para la gestión y control de proyectos.
- Se utiliza herramientas de administración de la configuración e integración continua.

Tabla XXI. **Ficha de evaluación 1**

HOJA DE EVALUACION

FECHA: _____ CICLO: _____

CURSO: _____

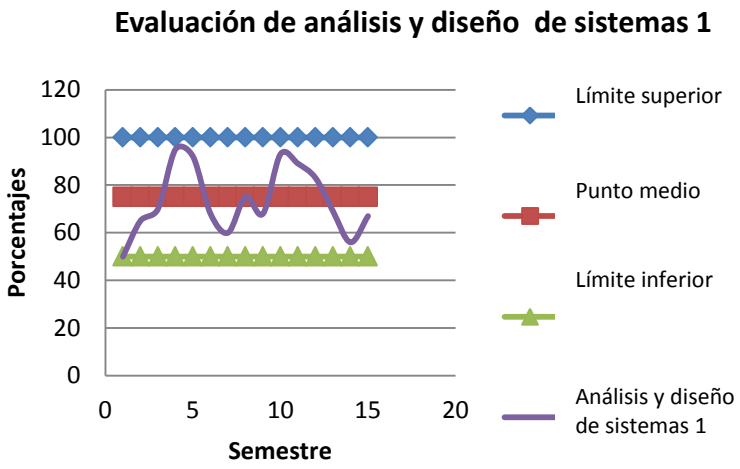
	ANALISIS Y DISEÑO DE SISTEMAS 1	ANALISIS Y DISEÑO DE SISTEMAS 2	SOFTWARE AVANZADO	IDEAL
CALIDAD DEL PROCESO DE DESAROLLO				10/100
COMPLETACION EN TIEMPO ESTIMADO				10/100
CUMPLIMIENTO DE LOS REQUERIMIENTOS				10/100
COMPROBACION DE LAS ESTIMACIONES CON LA EVALUACION REAL				10/100
MENOR INCIDENCIA DE FALLOS EN EL SISTEMA				10/100
MENOR INCIDENCIA DE FALLOS EN EL PROCESO				10/100
UTILIZA CORRECTAMENTE UNA METODOLOGIA DE DESARROLLO, (SI=10,NO=0)				10/100
IMPLEMTENTA BUENAS PRACTICAS DE DESARROLLO DE SOFTWARE, (SI=10,NO=0)				10/100
UTILIZA UN SISTEMA AUTOMATIZADO DE CONTROL DE PROYECTOS, (SI=10,NO=0)				10/100
UTILIZA HERRAMIENTAS DE ADMINISTRACION DE LA CONFIGURACION E INTEGRACION CONTINUA **cada una tiene una ponderacion de 5.*				10/100
TOTAL				100/100

Fuente: elaboración propia.

5.1.2. Gráfica de control

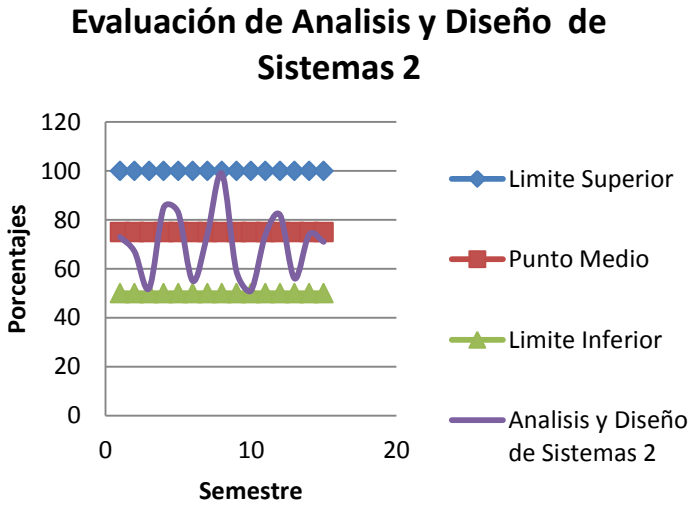
Con base en los resultados obtenidos en las hojas de evaluación, se procede a construir gráficos de control que faciliten la evaluación, como se muestra en los ejemplos:

Figura 49. **Gráfica de control 1**



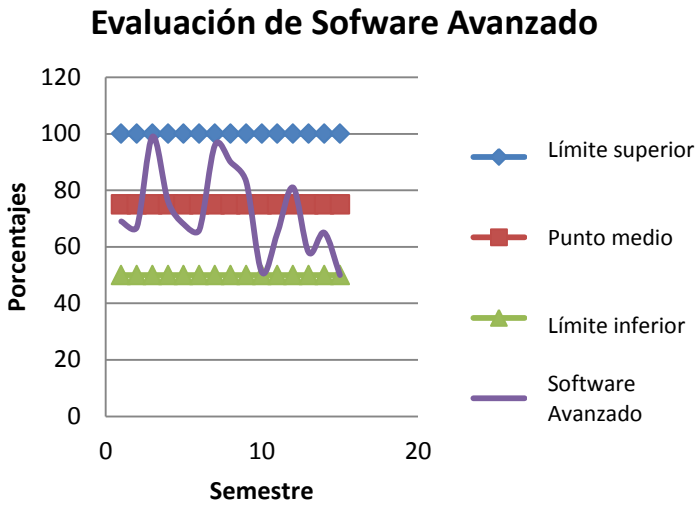
Fuente: elaboración propia.

Figura 50. **Gráfica de control 2**



Fuente: elaboración propia.

Figura 51. **Gráfica de control 3**



Fuente: elaboración propia.

Los ejemplos han sido creados con valores aleatorios, pero es posible utilizar los ejemplos para señalar que no debe de existir cambios tan drásticos entre un ciclo y otro, esto significaría que la calidad decae en ciertos semestres, lo cual debe de verificarse y buscar la causa de dicho problema.

Lo normal es que el comportamiento sea un poco constante y se esperaría que este fuera con una tendencia de crecimiento hacia el límite superior de la gráfica.

5.2. Estrategias para la implementación del cambio dentro del proceso de enseñanza

Para lograr la implementación de buenas prácticas, debe de establecerse dentro del proceso educativo una estrategia integral que facilite el proceso tanto

de enseñanza como de aprendizaje; durante el cual se relacionan el aprendizaje con los intereses, necesidades, experiencias y perspectivas profesionales del alumnado.

Para lo cual se recomienda seguir los siguientes pasos:

- El profesor no debe de estar solo
- Tener objetivos muy claros con respecto al proceso didáctico y marcar retos de aprendizaje.
- El profesorado debe de estar motivado e implicado en todo aquello que se trate de transmitir, fundamentalmente en el cambio y los objetivos del mismo.
- Conseguir que el alumnado protagonice el proceso de aprendizaje y confíe en sí mismo, demostrándole interés por sus iniciativas.
- Apropiarse del mapeo de objetos de aprendizaje desarrollado en el capítulo anterior, es decir el desconocimiento que se quiere insertar dentro del aprendizaje del proceso de desarrollo de software y su lugar dentro del proceso de asimilación y aprendizaje.
- Realizar una planificación consensuada de la aplicación e implementación de los nuevos objetos de aprendizaje.
- Desarrollar un proceso de evaluación, co evaluación y autoevaluación en el que todos los elementos personales implicados en el proceso didáctico colaboren en la mejora continua.
- Plantear nuevos modelos de evaluación dirigidos no sólo a los resultados sino también a los procesos, a las competencias, capacidades y todos los procesos relacionados con el trabajo realizado por el alumnado.

.

CONCLUSIONES

1. Según la investigación realizada y con los datos de la encuesta a egresados de Ingeniería en Ciencias y Sistemas, el fuerte de la carrera es precisamente el desarrollo de software, por lo cual se convierte en el punto crítico de control para ser evaluado dentro de los términos de calidad y considerarse importante la mejora continua del mismo como un objetivo de la Escuela de Ciencias y Sistemas.

2. Los problemas dentro del proceso de desarrollo de software de los estudiantes de la carrera de Ingeniería en Ciencias y Sistemas, giran en torno a 5 grandes temas que representan una debilidad para el proceso de desarrollo:
   - La administración de proyectos.
   - El control y la calidad de los proyectos.
   - La mala administración del tiempo.
   - La poca importancia que se le da al análisis de los requerimientos del sistema.
   - La falta de incorporación de técnicas probadas para un buen desarrollo de software, como lo es las buenas prácticas de desarrollo de software, los patrones de diseño y la utilización de una metodología de desarrollo adecuada y de forma correcta.

3. La aplicación de las prácticas presentadas en este trabajo, dentro del proceso de desarrollo, son de gran ayuda para palear las debilidades encontradas durante el diagnóstico; debilidades en la administración de proyectos, la baja calidad de los productos de software, la falta de

automatización en procesos de control y seguimiento, así como en la administración y la integración de módulos. Las buenas prácticas propuestas colaboran a mejorar la estimación de recursos a lo largo de los proyectos de desarrollo de software.

4. Para fortalecer las habilidades de programación y lograr un mejor rendimiento en los estudiantes durante los cursos de ingeniería de software, se concluye que es necesario reforzar el proceso de enseñanza y control del rendimiento de los estudiantes por medio de instrumentos como los que se proponen en este trabajo de investigación.

5. No existe una receta para un desarrollo perfecto, solo existen técnicas para reducir los riesgos durante el proceso de desarrollo; tanto en la gestión del proyecto como en la realización del software como tal. Durante el presente trabajo de investigación se sugiere técnicas y prácticas que fortalezcan y mejoren el proceso de desarrollo y los resultados de este tengan el valor agregado que se espera que tengan.

6. No existe una metodología que sea la mejor en todos los sentidos de la ingeniería de software, los criterios que deben de seguirse para elegir una metodología dentro de la amplia variedad que existe son:

   - Los requerimientos del sistema.
   - Las circunstancias específicas en las que se da el proyecto.

   Con base en estos 2, se debe de elegir una metodología de vanguardia que cumpla con las necesidades que se tienen dentro del proyecto.

7. La falta de humildad es el reto más grande para el aprendizaje y la mejora continua, como ingenieros existe una debilidad muy grande que consiste en la falta de humildad, lo cual afecta en muchas de las actividades de la ingeniería y es de esperarse que también en la ingeniería de software; durante el proceso de desarrollo de un sistema, la humildad puede ser la clave para identificar mejor los requerimientos de un sistema, si no se pretende saber todo, será de gran ayuda poder escuchar y preguntar, investigar al respecto.

   En el caso de la construcción de software se hace no solo con base en los conocimientos adquiridos de uno de los miembros del equipo, si no que con base en la estrategia más conveniente para el equipo. En resumen, la falta de humildad puede ser el factor que haga dejar de aprender nuevas cosas y cerrar la oportunidad a la mejora continua que cualquier proceso debe de buscar.

8. Si se logra automatizar bien la ejecución de las pruebas, se puede obtener métricas que ayuden a tener una retroalimentación tangible y comprobable de la evolución de la calidad del producto final del proyecto, como del proceso de construcción del mismo teniendo como herramientas históricos de las evaluaciones realizadas.

RECOMENDACIONES

1. Uno de los fuertes de la carrera de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala, es la ingeniería de software, por lo que se recomienda fortalecer la enseñanza a programar en la carrera, porque es en los primero cursos de la carrera donde los estudiantes adquieren las habilidades tanto de abstracción como de programación y estos deben ser la base para poder adquirir los demás conocimientos de ingeniería de software.

2. Utilizar las herramientas de evaluación propuestas en el capítulo 6 para el seguimiento del proceso de desarrollo de software, ya que con estas se puede tener una retroalimentación constante y comprobar en la línea del tiempo la efectividad del cambio, y pensar ampliar la evaluación a otras áreas de la carrera en el futuro.

3. Se sugiere incorporar dentro del proceso de enseñanza los objetos de aprendizaje identificados en el capítulo 4, con el fin de brindar a los estudiantes de la carrera las herramientas necesarias para un desarrollo de calidad y tener la valoración deseada en el ámbito profesional.

4. Modelar software con elementos visuales, esto colabora en la comprensión de los requerimientos y la solución a realizarse, para ello se recomienda utilizar diagramas de casos de uso, diagramas de clases, diagramas de estados, diagramas de componentes, diagramas de implementación.

El modelado visual eleva el nivel de abstracción para poder llevar la solución a código, pudiendo entender por medio de la abstracción de clases, subsistemas, entre otros. Esto ayuda a reforzar la recomendación anterior de fortalecer la ingeniería de software.

5. Controlar los cambios que se hacen al software, consiste en establecer actividades de control, registro de las mismas y monitoreo de cambios que faciliten el desarrollo iterativo. Se sugiere establecer entornos de trabajo seguro y confiable, lo cual significa que cada miembro del equipo debe utilizar prácticas como la administración de la configuración; además debe de considerarse la automatización de la integración y la administración de los *releases*.

6. La evaluación del software por medio de pruebas es una de las buenas practicas indispensables dentro del proceso y debe de evaluar continuamente la calidad de un sistema con respecto a funcionalidad, confiabilidad, performance, desarrollo y hasta la misma implementación , además de validar aspectos de gestión como tiempos y otras métricas; con el fin de detectar posibles causas de error o riesgo, ya que encontrar y reparar un error al final del proceso, puede ser de gran impacto en tiempo y en general en el producto final que implique pérdida para el estudiante.

   Se recomienda de la misma manera el uso del TDD para la evaluación puntual del sistema.

   La evaluación del estado del proyecto debe ser objetiva, se evalúan resultados de *test*. Se exponen inconsistencias en requerimientos, diseños e implementaciones.

Se focaliza esta en las áreas de mayor riesgo, esto es una práctica preventiva, la cual puede automatizarse y asegurar de esa manera en un alto grado la calidad de los resultados.

7. Fortalecer aspectos de administración de proyectos, así como el trabajo en equipo con la finalidad de que a los estudiantes les sea más fácil integrarse a equipos de trabajo y sean elementos de mayor valor dentro de sus equipos de trabajo. Aspectos como la comunicación y la coordinación son tan importantes y pueden llegar a convertirse en las habilidades más apreciadas dentro del trabajo en equipo.

8. La poca integración de los estudiantes de la carrera en los equipos de trabajo se debe sobre todo a la falta de confianza en las capacidades, tanto en sí mismos como en los demás integrantes del equipo; por lo que se recomienda además iniciar los proyectos grupales desde el inicio del aprendizaje de la ingeniería de software.

BIBLIOGRAFÍA

1. ALBARRACÍN BELTRÁN, Maryory. *Profundización en ingeniería de software*. Universidad Pedagógica y tecnológica de Colombia UPTC. 2013. [en línea] http://es.scribd.com/doc/168670984/ Metodologías-Tradicionales. [Consulta: mayo de 2013]

2. BRAUDE, Eric; BERNSTEIN, Michael. *Software engineering: Modern approaches.* 2a ed*.* USA: Wiley & Sons Incorporated. 2010. 800 p. ISBN-13: 978-0471692089

3. *EVALUACIÓN DEL PÉNSUM DE ESTUDIOS DE INGENIERÍA EN CIENCIAS Y SISTEMAS, FACULTAD DE INGENIERÍA, USAC, RESPECTO DEL PERFIL DEL EGRESAD*' Facultad de Ingeniería, Universidad de San Carlos de Guatemala. No. 53963. [Consulta: marzo de 2013]

4. FITZGERALD, Briam. *An empirical investigation into the adoption of systems develpement methodologies, information & management.* [en línea]. http://www.brian-fitzgerald.com/wp-content/uploads/2013 /10/InfoMgt.pdf. [Consulta: junio de 2013].

5. HERNÁNDEZ, Roberto; FERNÁNDEZ, Carlos; BAPTISTA, Pilar. *Metodología de la investigación.* 3a. ed. México: McGraw-Hill. 2003. 689 p. [en línea]. http://data.over-blogkiwi.com/0/27/01/47/201304 /ob_195288_metodologia-de-la-investigacion-sampieri-ernande.pdf. [Consulta: mayo de 2013]

6. MARTIN, Robert. *Código limpio*. Massachusetts: Anaya Multimedia. 2002. 431 p. ISBN 0-13-235088-2

7. MCLAUGHLIN, Brett; POLLICE, Gary; WEST, Dave. *Head First Object-Oriented Analysis and Design.* Boston: O'Reilly Media, 2006. 636 p.

8. *Metodologías Tradicionales:* metodologías ágiles. Daniel Zs. 2012. [en línea]. http://danielzs75.blogspot.com/2012/10/metodologias-agiles-extreme-programming.html. [Consulta: 1 de mayo de 2013].

9. ORTIZ, N. *La elaboración de los proyectos de investigación.* [en línea]. http://www.monografias.com/trabajos/elabproy [Consulta: 1 de febrero 2013]

10. 7th ANNUAL STATE of AGILE DEVELOPMENT SURVEY. VERSIONONE. 2013. No 7. [en línea] http://www.versionone.com /pdf/7th-Annual-State-of-Agile Development- Survey. [Consulta: 1 de mayo de 2013].

APÉNDICES

Fichas resumen de cursos

Apéndice 1. Introducción a la Programación y Computación 1

Facultad de
Ingeniería
Escuela de Ciencias y Sistemas
Ficha de información de curso

Nombre	Introducción a la Programación y Computación 1
Código	770
Créditos	4
Capítulos	8
Epígrafes	49
Sub epígrafes	82
Índice de carga académica	46,33 por ciento
Metodología	Clases diarias.
	Elaboración de investigaciones y tareas.
	Práctica de exámenes cortos y parciales.
	Laboratorio taller.
	Elaboración de proyectos de programación.
Objetivo general	Lograr que el estudiante adquiera la habilidad de programar y los conocimientos básicos de la programación utilizando el paradigma orientado a objetos.
Objetivos específicos	Integrar al estudiante a la tecnología de la computación.
	Conocer las diferentes metodologías de programación.
	Organizar soluciones utilizando un lenguaje de programación.
	Adquirir la habilidad de hacer algoritmos.
	Aprender a elaborar diseños de clases preliminares en UML.
	Analizar los problemas con metodología orientada a objetos.
	Conocer el lenguaje Java como el primer lenguaje de programación para computadoras.
Fuente	http://caecys.org/index.php/estudiante/listdocumentos/slug/1

Fuente: elaboración propia.

Apéndice 2. Introducción a la Programación y Computación 2

Facultad de
ingeniería
Escuela de Ciencias y Sistemas
Ficha de información de curso

Nombre	Introducción a la Programación y Computación 2
Código	771
Créditos	5
Capítulos	5
Epígrafes	30
Sub epígrafes	132
Índice de carga académica	55,67 por ciento
Metodología	El curso se impartirá a través de clases magistrales de 4 períodos semanales impartidos dos días por semana. Cada día 2 periodos.
	El laboratorio se impartirá una vez por semana, con duración de 2 períodos cada día.
	Durante el semestre, se asignarán 3 proyectos de programación, a realizarse de manera individual; así como tareas, ejercicios e investigaciones.
Objetivo general	Preparar al estudiante para desarrollar aplicaciones de software utilizando un enfoque orientado a objetos.
Objetivos específicos	Que el estudiante modele los problemas de una forma estándar y profesional.
	Que el estudiante logre un mayor proceso de abstracción en los problemas que resuelva.
	Que el estudiante utilice una metodología para desarrollar aplicaciones de software.
Fuente	http://caecys.org/index.php/estudiante/listdocumentos/slug/1

Fuente: elaboración propia.

Apéndice 3. Estructuras de Datos

Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Ficha de información de curso

Nombre	Estructuras de Datos
Código	772
Créditos	5
Capítulos	6
Epígrafes	22
Sub epígrafes	0
Índice de carga académica	9,33 por ciento
Metodología	Durante el curso se programaran clases presenciales con contenido audiovisual, lecturas, tareas e investigaciones para hacer en el horario de clase y en casa.
Objetivo general	Proporcionar al estudiante los conocimientos y prácticas necesarias acerca de las estructuras de datos y algoritmos de manipulación de las mismas, enfocándose en un ambiente de desarrollo web.
Objetivos específicos	Conocer el funcionamiento y saber escribir algoritmos que manipulan las estructuras de datos más utilizadas.
	Utilizar la metodología orientada a objetos para la solución de problemas que utilizan las diferentes estructuras de datos, independientemente de cualquier lenguaje de programación.
	Implantar las diferentes estructuras de datos estudiadas en la computadora utilizan un lenguaje de programación y siguiendo la metodología orientada a objetos para ambientes web.
Fuente	http://caecys.org/index.php/estudiante/listdocumentos/slug/1

Fuente: elaboración propia.

Apéndice 4. Manejo e Implementación de Archivos

Universidad
de San
Carlos de
Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Ficha de información de curso

Nombre	Manejo e Implementación de Archivos
Código	773
Créditos	4
Capítulos	4
Epígrafes	24
Sub epígrafes	64
Índice de carga académica	30,67 por ciento
Metodología	Clases magistrales, complementado con auto estudio por parte del estudiante así como apoyo magistral y práctico del curso.
Objetivo general	Se busca que los estudiantes puedan comprender de forma práctica y teórica los aspectos generales en relación al manejo de información y su almacenamiento físico, tanto a nivel básico como al de un sistema administrador de bases de datos.
Objetivos específicos	Tener una base sólida del concepto y uso de archivos en un sistema de información.
	Tener un amplio concepto de lo que son las bases de datos, la forma en que funcionan y sus principales características.
	Adquirir los conocimientos necesarios para poder administrar una base de datos tecnológicamente actualizada.
Fuente	http://caecys.org/index.php/estudiante/listdocumentos/slug/1

Fuente: elaboración propia.

Apéndice 5. Sistemas de Bases de Datos 1

Ficha de información de curso

Código	774
Créditos	5
Capítulos	3
Epígrafes	15
Sub epígrafes	0

Índice de carga académica	6,00 por ciento
Metodología	El contenido del laboratorio será impartido a través de clases magistrales en donde se presentaran los temas y conceptos que serán de utilidad para que el estudiante pueda conocer, comprender, entender, finalmente lograr aplicarlos en cada una de las actividades que se realicen a lo largo del desarrollo del curso, el material didáctico será publicado para apoyo del estudiante.
Objetivo general	Se busca introducir al estudiante en cada uno de los procesos necesarios para el desarrollo e implementación de una base de datos, desde la concepción del modelo de datos, instalación y configuración del DBMS, hasta la creación de reportes para toma de decisiones.
Objetivos específicos	Que el estudiante sea capaz de realizar un modelo de datos optimo basado en las normas formales
	Guiar al estudiante en la instalación y configuración de un DBMS
	Que el estudiante conozca buenas prácticas de mantenimiento de un DBMS
	Que el estudiante sea capaz de elaborar consultas básicas, intermedias y avanzadas, para presentación de datos
	Introducir al estudiante a Reporting Services
Fuente	http://caecys.org/index.php/estudiante/listdocumentos/slug/1

Fuente: elaboración propia.

Apéndice 6. Sistemas de Bases de Datos 2

Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Ficha de información de curso

Nombre	Sistemas de Bases de Datos 2
Código	775
Créditos	5
Capítulos	7
Epígrafes	61
Sub epígrafes	0
Índice de carga académica	22,67 por ciento
Metodología	El curso se desarrollará intercalando clases magistrales para la exposición de conceptos nuevos y clases participativas en las que el estudiantes puede compartir lo investigado por su cuenta.
Objetivo general	Conocer y aplicar la teoría que fundamenta el funcionamiento de los sistemas administrativos de bases de datos.
Objetivos específicos	Conozca y aplique los conceptos que fundamentan la concurrencia en un sistema administrador de base de datos.
	Aplique los conceptos y procedimientos de respaldo y recuperación de bases de datos.
	Implemente los diferentes modelos de bases de datos.
	Entienda los sistemas actuales y su funcionamiento basados en conceptos teóricos.
Fuente	http://caecys.org/index.php/estudiante/listdocumentos/slug/1

Fuente: elaboración propia.

Apéndice 7. Análisis y Diseño de Sistemas 1

Universidad
de San
Carlos de
Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Ficha de información de curso

Nombre	Análisis y Diseño de Sistemas 1
Código	283
Créditos	4
Capítulos	4
Epígrafes	16
Sub epígrafes	25
Índice de carga académica	15,00 por ciento
Metodología	Clase magistral impartida 2 veces por semana
	Proyecto de modelado de negocio y especificación de requerimientos
	Tareas y evaluaciones complementarias
	Autoestudio
Objetivo general	
Objetivos específicos	Entender que es una metodología de desarrollo de software y su contexto en la ingeniería de software
	Conocer las principales metodologías de desarrollo de software aplicadas en la actualidad
	Entender y desarrollar modelos de procesos de negocio
	Entender y desarrollar métodos para la definición de requerimientos para construcción de sistemas informáticos
Fuente	http://caecys.org/index.php/estudiante/listdocumentos/slug/1

Fuente: elaboración propia.

Apéndice 8. Análisis y Diseño de Sistemas 2

Escuela de Ciencias y Sistemas
Ficha de información de curso

Código/créditos **Capítulos** **Epígrafes** **Sub epígrafes** **Índice de carga académica**	785/4 7 28 10 15,00 por ciento
Metodología	Los conocimientos del laboratorio serán transmitidos por medio de exposiciones y demostraciones hechas en clases, así como la solución a problemas frecuentes y/o preguntas que se puedan generar al momento de la explicación.
	Todo envió de tareas, propuestas, etc., será con un asunto definido por el auxiliar el día que se solicite, de igual manera el formato y extensión, se utilizará como medio de comunicación y en algunos casos de entrega de tareas por medio del correo electrónico proporcionado por la auxiliar y en algunos casos una herramienta de administración de proyectos. Se solicitarán las fases del proyecto y otras tareas impresas o demostraciones físicas.
Objetivo general	Lograr que el estudiante adquiera los conocimientos necesarios para poder analizar y diseñar un sistema de acuerdo a las tecnologías y herramientas disponibles, adoptando para ello las buenas prácticas de diseño y de programación de sistemas
Objetivos específicos	Que el estudiante sea capaz de analizar requerimientos de un sistema utilizando artefactos y metodológicas de desarrollo de sistemas adecuadas.
	Que el estudiante entienda, sea capaz y practique la administración de la configuración e integración continua de un proyecto de software.
	Conocer y poner en práctica los conocimientos de arquitectura de software abarcados durante el curso.
	Profundizar la arquitectura orientada a servicios, las ventajas y facilidades que ofrece y poner en práctica las mismas.
	Que el estudiante sea capaz de diseñar, configurar, construir e implementar un proyecto de software empresarial.
	Que el estudiante sea capaz de diseñar un sistema a partir de los requerimientos tomando en cuenta las buenas prácticas de desarrollo.
	Que el estudiante sea capaz de documentar todo el proceso de desarrollo de sistemas bajo estándares aceptables dentro del mercado.
Fuente	http://caecys.org/index.php/estudiante/listdocumentos/slug/1

Fuente: elaboración propia.

Apéndice 9. Seminario de Sistemas 1

Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Ficha de información de curso

Nombre	Seminario de Sistemas 1
Código	797
Créditos	3
Capítulos	
Epígrafes	
Sub epígrafes	
Índice de carga académica	#¡DIV/0!
Metodología	Para el laboratorio del curso se programaran clases presenciales con contenido audiovisual, lecturas, y prácticas para hacer en el horario del laboratorio y en casa, como también investigación y presentación de los contenidos.
Objetivo general	Lograr que el estudiante conozca las tendencias en cuanto a plataformas y metodologías actuales del área de IT, investigando y definiendo las características de cada una de estas tecnologías.
Objetivos específicos	Conocimiento de las herramientas de gestión de información más importantes hoy en día en las empresas.
	Conocimiento de las tendencias de las tecnologías de información que están cambiando la forma de la gestión de la información y las herramientas
	Descripción general de desarrollo profesional en el área de informática
Fuente	http://caecys.org/index.php/estudiante/listdocumentos/slug/1

Fuente: elaboración propia.

Apéndice 10. Seminario de Sistemas 2

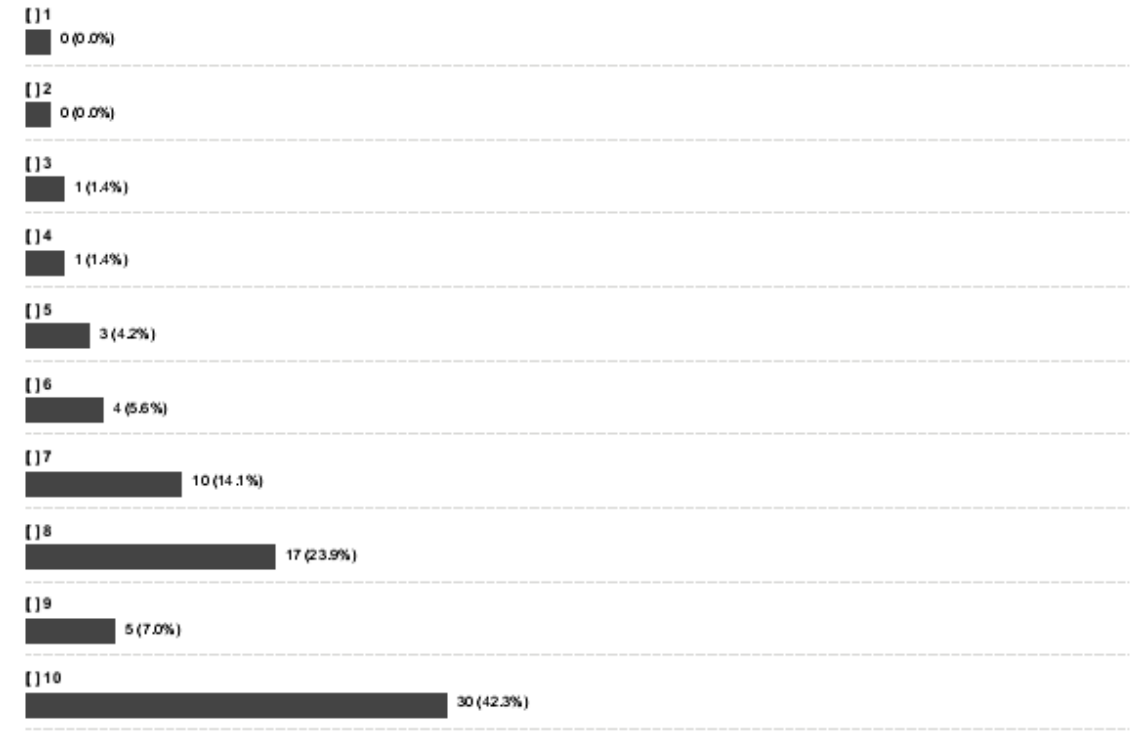
Universidad
de San Carlos
de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Ficha de información de curso
Autora: Teresa Domicila Quezada Mauricio

Nombre	Seminario de Sistemas 2
Código	798
Créditos	3
Capítulos	5
Epígrafes	0
Sub epígrafes	0
Índice de carga académica	1,67 por ciento
Metodología	Exposición teórica de los conceptos de Business Intelligence en 3 periodos semanales, ayudados de casos prácticos que se dan en las empresas. Clase magistral , Prácticas de laboratorio y proyectos donde se apliquen las técnicas aprendidas en clase. Evaluaciones cortas, parciales y una final.
Objetivo general	Complementar la formación de un profesional en Inteligencia de Negocios (BI – Business Intelligence), con habilidades para el manejo de las principales técnicas y herramientas en el ramo y obtener la experiencia necesaria implementando varias aplicaciones, así como la consideración de escenarios por medio de tableros de decisión para diferentes modelos comerciales de negocios.
Objetivos específicos	Reunir, depurar y transformar todos los datos que la empresa almacena en información estructurada y coherente.
	Aplicar, analizar y convertir la información obtenida en conocimiento que ayude en la toma de decisiones estratégicas y operacionales.
	Comprender y explicar las razones para emprender un proyecto de Business Intelligence en alguna organización.
Fuente	http://caecys.org/index.php/estudiante/listdocumentos/slug/1

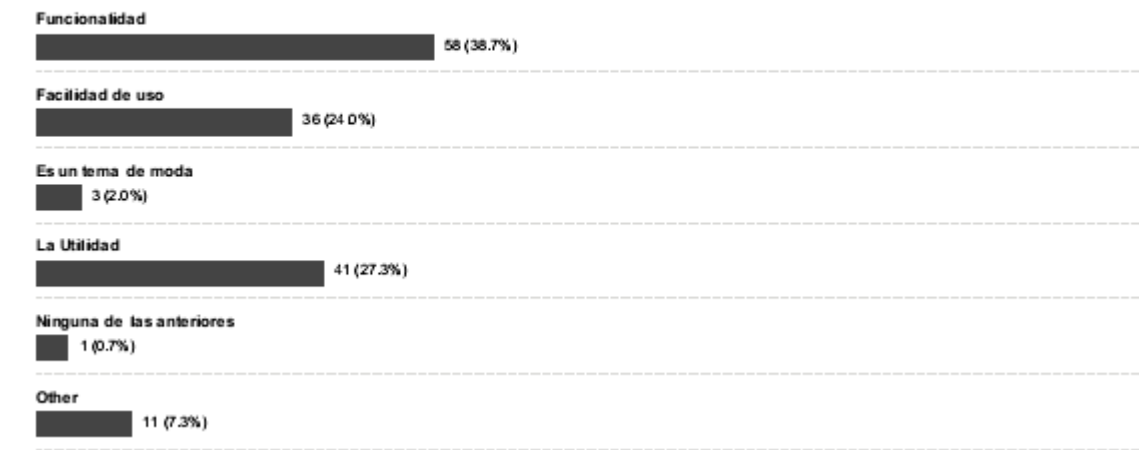
Fuente: elaboración propia.

Apéndice 11. Resultados de encuesta a estudiantes

1) En una escala del 1 al 10, dónde 10 es "muy interesante" y 1 es "nada interesante" ¿Qué tan interesante le parece el tema "Las buenas Prácticas de Desarrollo de Software" ?



2) ¿Cuál o cuáles de las siguientes características le atraen del tema de implementación de buenas prácticas en el desarrollo de software?



Continuación del apéndice 11.

3) ¿En qué cursos de la carrera le han enseñado acerca de buenas prácticas para el desarrollo de software?

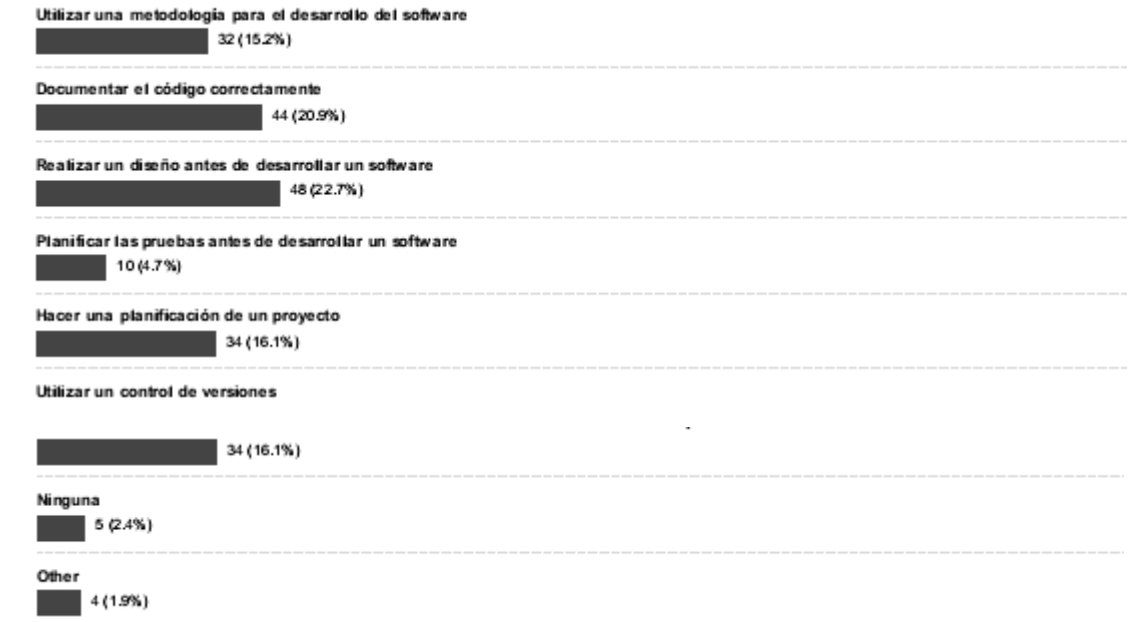
	Nada	Poco	Regular	Mucho	SNE	Responses	Average Score
IPC1 e IPC2	31 (43.66%)	27 (38.03%)	10 (14.08%)	3 (4.23%)	0 (0.00%)	71	1.79 / 5(35.80%)
Manejo e implementación de Archivos	36 (50.70%)	28 (39.44%)	7 (9.86%)	0 (0.00%)	0 (0.00%)	71	1.59 / 5(31.80%)
Estructuras de Datos	27 (38.03%)	17 (23.94%)	15 (21.13%)	12 (16.90%)	0 (0.00%)	71	2.17 / 5(43.40%)
Sistemas de Bases de Datos 1 y Sistemas de Bases de Datos2	21 (29.58%)	19 (26.76%)	21 (29.58%)	9 (12.68%)	1 (1.41%)	71	2.30 / 5(46.00%)
AyD1 yAyD2	1 (1.41%)	7 (9.86%)	29 (40.85%)	31 (43.66%)	3 (4.23%)	71	3.39 / 5(67.80%)
Seminario de Sistemas 1 ySeminario de Sistemas 2	27 (38.03%)	22 (30.99%)	16 (22.54%)	2 (2.82%)	4 (5.63%)	71	2.07 / 5(41.40%)
Software Avanzado	4 (5.63%)	13 (18.31%)	12 (16.90%)	9 (12.68%)	33 (46.48%)	71	3.76 / 5(75.20%)
							2.44 / 5 (48.77%)

4) ¿En qué cursos se debería de reforzar la enseñanza de buenas prácticas para el desarrollo de software?

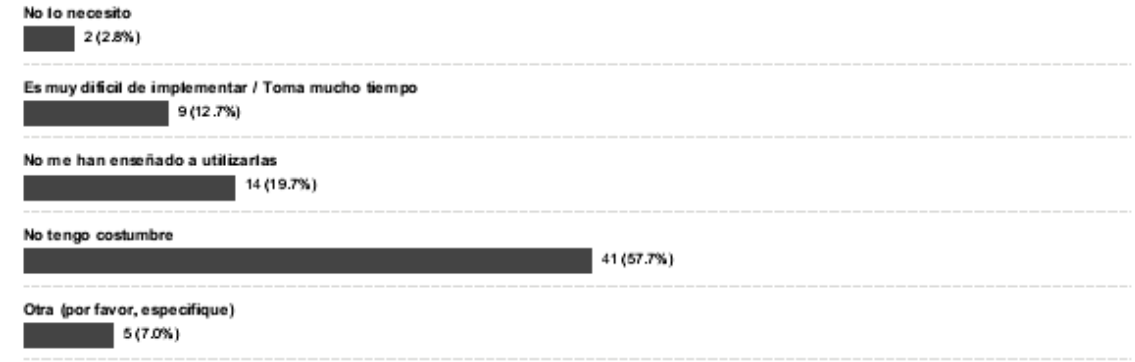
	Nada	Poco	Regular	Mucho	SNE	Responses	Average Score
IPC1 e IPC2	1 (1.41%)	3 (4.23%)	13 (18.31%)	54 (76.06%)	0 (0.00%)	71	3.69 / 5(73.80%)
Manejo e implementación de Archivos	5 (7.04%)	13 (18.31%)	29 (40.85%)	24 (33.80%)	0 (0.00%)	71	3.01 / 5(60.20%)
Estructuras de Datos	5 (7.04%)	14 (19.72%)	27 (38.03%)	24 (33.80%)	1 (1.41%)	71	3.03 / 5(60.60%)
Sistemas de Bases de Datos 1 y Sistemas de Bases de Datos2	5 (7.04%)	12 (16.90%)	32 (45.07%)	22 (30.99%)	0 (0.00%)	71	3.00 / 5(60.00%)
AyD1 yAyD2	5 (7.04%)	3 (4.23%)	15 (21.13%)	47 (66.20%)	1 (1.41%)	71	3.51 / 5(70.20%)
Seminario de Sistemas 1 ySeminario de Sistemas 2	3 (4.23%)	11 (15.49%)	37 (52.11%)	18 (25.35%)	2 (2.82%)	71	3.07 / 5(61.40%)
Software Avanzado	0 (0.00%)	5 (7.04%)	13 (18.31%)	38 (53.52%)	15 (21.13%)	71	3.89 / 5(77.80%)
							3.31 / 5 (66.29%)

Continuación del apéndice 11.

5) De las siguientes buenas prácticas... ¿Cuáles son las que comúnmente utiliza para el desarrollo de software?



6) ¿Cuál o cuáles de las siguientes características reflejan de alguna manera su realidad en cuanto a la implementación de las buenas prácticas de desarrollo de software?



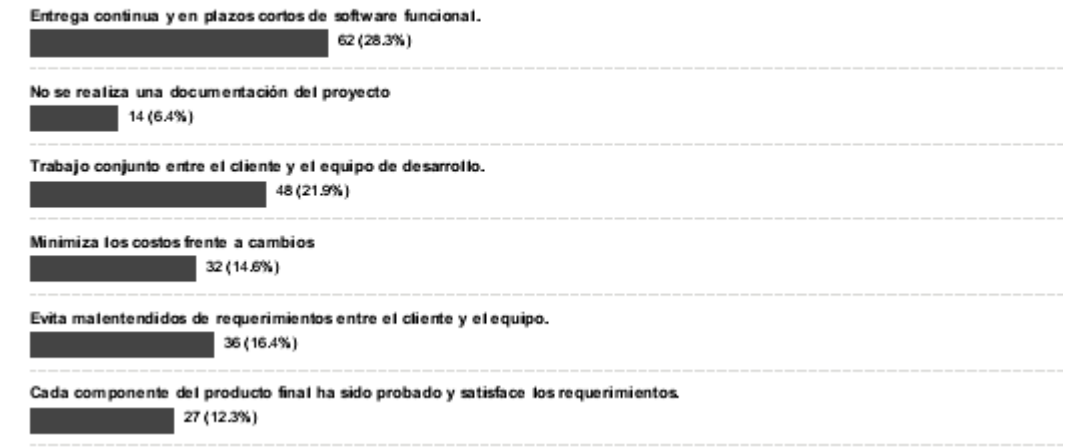
Continuación del apéndice 11.

7) ¿Cuál o cuáles de las siguientes prácticas le parece importante utilizar para el desarrollo de software?

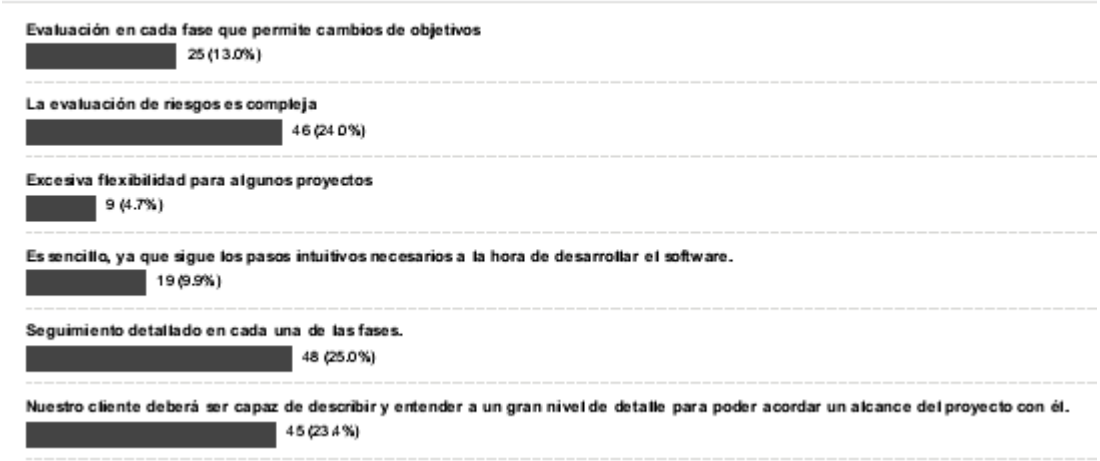
8) ¿Creé que la aplicación integrada de las prácticas, en el contexto de una metodología de desarrollo, debe de ser parte del contenido y prácticas de los cursos de desarrollo de software?

Continuación del apéndice 11.

9) ¿Cuáles de las siguientes características creé usted que describen una metodología ágil?

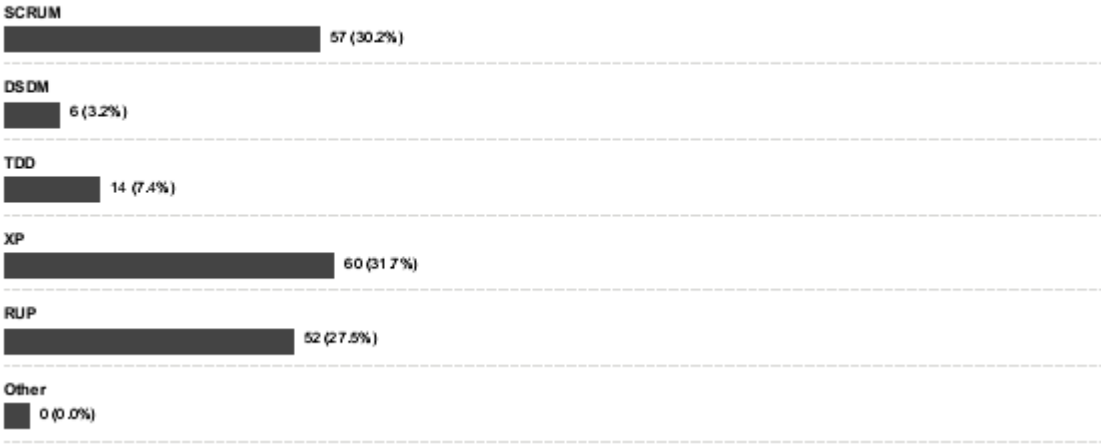


10) ¿Cuáles de estas características corresponden a una metodología tradicional o rígida?



Continuación del apéndice 11.

11) ¿Qué metodología(s) de software considera usted que deben de enseñarse y aplicarse en los distintos años de la carrera?



12) ¿Tiene algún comentario o sugerencia para mejorar la calidad de software que se realiza durante los diferentes años de la carrera?

Fuente: elaboración propia.