

00623



FACULTAD DE CONTABILIDAD
Y ADMINISTRACION

AGO. 30 2004



COORDINACION DE
EXAMENES PROFESIONALES



Universidad Nacional
Autónoma de México

Dirección General de Bibliotecas de la UNAM

Biblioteca Central



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

“METODOLOGÍAS DE DESARROLLO DE SOFTWARE COMO
HERRAMIENTA PARA ASEGURAR LA CALIDAD EN LOS
PROYECTOS INFORMÁTICOS”

TESIS PROFESIONAL
QUE PARA OBTENER EL TÍTULO
DE LICENCIADO EN INFORMÁTICA
PRESENTAN:
MA. GUADALUPE BAUTISTA ZARAGOZA
MARGARITA GARIBAY TIERRADENTRO

ASESOR:
DR. RICARDO ANTONIO RIVERA SOLER



MÉXICO. D. F.

2004

AGRADECIMIENTOS

Es difícil manifestar los sentimientos en que me halló en estos momentos, no encuentro palabras, sin embargo, se que si se descubren en las próximas líneas sabrán lo que siento, aunque lo que haya escrito no lo traduzca de esa manera.

Gracias:

A mis padres por su amor

...yo también los amo, son realmente nobles y buenas personas

A lo que yo llamo mi segunda casa, la Universidad Nacional Autónoma de México porque me abrió sus puertas

...dejo que entrara, y me alimentó en conocimientos...en alma

A mi asesor Dr. Ricardo Rivera Soler por su guía

...para mí resulta excelente, me ayudó, y ver en su persona esa gran calidad humana me hace reflexionar

A mis profesores por sus enseñanzas

...me otorgaron fuertes cimientos en el saber y me inyectaron esa fuerza en el querer conocer aún más

A Guadalupe porque recorrimos juntas este camino

...de no haber sido así, hubiera resultado difícil, hicimos excelente equipo

A mis hermanos por su apoyo

...sin él no lo hubiera logrado, he crecido junto a ellas y también las amo por lo que son

A mis amigos porque caminan a mi lado

...espero que no dejen de hacerlo

Margarita

AGRADECIMIENTOS

Gracias a:

Dios: *Por existir*

Mis padres: *Por su ejemplo de rectitud y gran calidad humana, son y serán el motivo de mi más grande orgullo.*

Mis profesores: *Por el preciado legado recibido de sus conocimientos y porque supieron inculcarme el sentido de responsabilidad, trabajo y disciplina.*

Mi asesor: Dr. Ricardo Antonio Rivera Soler
Por todo el conocimiento y el gran apoyo recibido en la realización de esta tesis.

Mi compañera de tesis: Margarita
Por contagiarme con su entusiasmo y dedicación.

Mis hermanos: *Por su cariño, comprensión y apoyo incondicional.*

Mis sobrinos: *Por su amor, inocencia y alegría.*

Mis mejores amigos: *Con los que he tenido la fortuna de convivir en el transcurso de mi existir, por su entrega noble y desinteresada.*

En la Sudirección de Sistemas de DGSCA-UNAM: *En especial a la M. C. Marcela J. Peñaloza Báez por darme la oportunidad de colaborar en esta gran institución, por el apoyo recibido en la realización de esta tesis a: L. I. María Teresa Ventura Miranda, Lourdes Hernández López y a los entusiastas chico y chicas DES.*

Con cariño y respeto

Ma. Guadalupe

DEDICATORIAS

A mis Padres: Carmen y Pedro

Como testimonio de gratitud ilimitada, porque me han apoyado con su amor y confianza, dejando tal vez que yo aprendiera de mis errores y aciertos, pero siempre con el respaldo de ustedes, así es como hemos logrado uno de los anhelos más grandes de mi vida, el de terminar mis estudios profesionales que constituyen el legado más grande que pudiera recibir y por el cual les viviré eternamente agradecida.

Ma. Guadalupe

A mis Padres: Teresa y Rosalio

Me siento afortunada de que ustedes siempre trabajaron muy duro para que yo pudiera recibir educación, ya que con ello no sólo me abrieron paso a algo mejor, sino que pude descubrir un mundo infinitamente maravilloso, ese ha sido y es el más grande sueño que yo estoy satisfecha, feliz de haberles cumplido.

Margarita

**A la Universidad Nacional Autónoma de México y en especial a la
Facultad de Contaduría y Administración**

Por brindarnos la oportunidad de prepararnos académicamente, incorporándonos a la sociedad como individuos activos al campo laboral y por integrarnos a la máxima casa de estudios basada en principios éticos y morales de la cual nos sentimos orgullosas y por la que seguimos luchando constantemente para contribuir a su grandeza.

MIL GRACIAS

"POR MI RAZA HABLARÁ EL ESPÍRITU"

INDICE

INDICE

INTRODUCCIÓN

1 MARCO CONTEXTUAL

1.1 Antecedentes.....	3
1.2 Identificación del problema.....	3
1.3 Demarcación del fenómeno.....	4
1.4 Conocimiento empírico en el medio.....	4
1.4.1 Nivel de las personas entrevistadas.....	4
1.4.2 Objetivo del cuestionario.....	4
1.4.3 Personas entrevistadas.....	5
1.4.4 Conclusión de cada pregunta.....	6
1.5 Opiniones profesionales.....	13
1.5.1 Nivel de las personas entrevistadas.....	13
1.5.2 Objetivo del cuestionario.....	13
1.5.3 Personas entrevistadas.....	14
1.5.4 Conclusión a cada pregunta.....	15
1.6 Conclusión a general.....	22
1.7 Hipótesis preliminar.....	22
1.8 Objetivos.....	22

2 MARCO TEÓRICO

2.1 Acopio de libros.....	25
2.1.1 Libros de estudio.....	25
2.1.2 Libros de lectura ligera.....	38
2.1.3 Libros de lectura rápida.....	45
2.2 Tesis.....	45
2.3 Conferencias.....	48
2.4 Diplomados.....	50

3 MARCO CONCEPTUAL

3.1 Antecedentes de las metodologías de desarrollo de software.....	51
3.1.1 Origen de las metodologías de desarrollo de software.....	51
3.2 Visión histórica de las metodologías de desarrollo de software.....	53
3.3 Fundamentos de las metodologías de desarrollo de software.....	61
3.3.1 Crisis del software.....	64
3.3.2 Ingeniería de software.....	65
3.3.3 Modelos de proceso.....	66
3.4 Monografía de las metodologías de desarrollo de software.....	74
3.4.1 Metodologías estructuradas.....	74
3.4.2 Metodologías orientadas a objetos.....	79
3.4.3 Metodologías de prototipo.....	84
3.4.4 Metodologías ágil de desarrollo o evolutiva.....	86
3.5 Definiciones de las metodologías de desarrollo de software.....	90
3.5.1 Definición etimológica.....	90
3.5.1 Definición de diccionarios.....	91

3.5.1 Definición de autores.....	92
3.5.1 Definición propia.....	92
3.5.1 Definición de sinónimos y antónimos.....	93
3.6 Evolución de las metodologías de desarrollo de software	94
3.7 Clasificación de las metodologías de desarrollo de software.....	96
3.8 Tendencias de las metodologías de desarrollo de software.....	98

4 MARCO METODOLÓGICO

4.1 Variables.....	101
4.2 Variables de control.....	101
4.3 Definición del universo.....	101
4.4 Determinación de la muestra.....	101
4.5 Determinación de la muestra.....	102
4.6 Costo de investigación	102
4.7 Cuestionario.....	102
4.8 Personas entrevistadas.....	103
4.9 Análisis de la información recopilada.....	104
4.10 Conclusión sobre los resultados.....	122
4.11 Aprobación de la hipótesis.....	123

5 MARCO INSTRUMENTAL

5.1 Propuesta de acción	125
5.2 Plan y programa de trabajo.....	126
5.2 Plan y programa de trabajo.....	126

CONCLUSIONES

Conclusiones del marco contextual.....	129
Conclusiones del marco teórico.....	130
Conclusiones del marco conceptual.....	131
Conclusiones del marco metodológico.....	132
Conclusiones del marco instrumental.....	132
Conclusiones generales.....	133

ANEXOS

A. Cuestionario para obtener conocimiento empírico.....	135
B. Recopilación de cuestionario.....	140
C. Cuestionario para obtener opiniones profesionales.....	143
D. Recopilación de cuestionario.....	148
E. Cuestionario final.....	152
F. Página de Internet.....	155
G. Carta de presentación para publicación de artículo.....	157
H. Artículo para publicación.....	158
G. Carta de presentación para publicación de artículo.....	157
I. Carta de presentación para propuesta de asignatura.....	162
J. Temario para materia optativa.....	163

GLOSARIO ACRÓNIMOS Y SIGLAS

BIBLIOGRAFÍA

165
171

INTRODUCCIÓN

Desde que nació la idea de los sistemas informáticos se ha afirmado que la medida, la disciplina, y un especial interés por la calidad darán lugar a un software que satisfaga las necesidades del usuario, que sea fiable, que se pueda mantener, en pocas palabras un software que sea mejor.

Como habría de esperarse, en las empresas pequeñas (consultoras de software) donde generalmente se desarrollan proyectos o sistemas informáticos pequeños, tienden a ser relativamente informales: los proyectos de desarrollo de sistemas nacen de conversaciones entre el usuario y el administrador del proyecto (que puede estar a su vez en cualquiera de las categorías de personas que se involucran en el desarrollo de software), y el desarrollo del sistema procede desde el análisis, diseño e implantación sin mayor alboroto.

Sin embargo, en las grandes empresas, en donde los proyectos tienden a ser de gran complejidad, todo se realiza de una manera mucho más formal, la comunicación entre los usuarios, la administración y el equipo del proyecto suele ser por escrito y se entiende que el proyecto pasará por diversas fases antes de completarse. Aún así es sorprendente ver la gran diferencia entre las maneras en que dos administradores afrontan sus respectivos proyectos.

Recientemente, ha empezado a cambiar el enfoque que se le da al desarrollo de sistemas, cada vez son más las organizaciones grandes y pequeñas que están adaptando una manera uniforme para el desarrollo de sus proyectos informáticos, esto a veces se conoce como plan de proyecto, metodología de desarrollo de software o simplemente "la forma en que hacemos las cosas".

Por la importancia que ha adquirido este tema en nuestro ámbito profesional, se vuelve prioritario estudiarlo, aprenderlo y practicarlo, así como consultar a los estudiosos del tema, para efectos de desempeñar un mejor papel como profesionistas en el área de desarrollo de software.

El presente trabajo de tesis tiene como objetivo estudiar las metodologías más utilizadas actualmente para el desarrollo de software, así como sus aspectos importantes para identificar la "metodología idónea" para cada tipo de proyecto que pueda cubrir las necesidades específicas. Con base en los conocimientos adquiridos mediante investigación temática en libros, tesis, conferencias, en Internet y entrevistas a usuarios de las metodologías. Hemos tratado de que los temas y las explicaciones sean claras, evitando crear confusión.

Este trabajo de tesis se divide en cinco capítulos:

- 1) Marco contextual
- 2) Marco teórico
- 3) Marco conceptual
- 4) Marco metodológico
- 5) Marco instrumental

En el primer capítulo *Marco contextual* presentamos el planteamiento del problema, tratando la importancia por la cual esta investigación se llevo a cabo, por medio de entrevistas a personas que conocen sobre el tema ya sea a nivel empírico o práctico. Se llegó a una hipótesis preliminar que grosso modo demuestra que la falta de conocimientos al elegir una metodología de desarrollo de software, provoca que el software se haga de manera impredecible e ineficiente, con lo que se crea la insatisfacción en el cliente.

El segundo capítulo denominado *Marco teórico* es una recopilación de información existente de nuestro tema de tesis (nuestra documentación). Además de formar parte del material para la realización del marco conceptual que incluye diversos conceptos y temas relacionados como son: gestión de proyectos, ciclo de vida de desarrollo de software, tipos de sistemas, análisis y diseño estructurado, análisis y diseño orientado a objetos y proceso unificado.

En el tercer capítulo *Marco conceptual* se analizan todos los conceptos que pueden tener relevancia en el tema central de ésta tesis, dando un seguimiento de esta temática desde su origen pasando por una visión histórica, que de una manera tajante intenta definir todos los antecedentes que pueda tener el tema central de este trabajo, desde el ábaco hasta la metodología de desarrollo de software más evolucionada que actualmente se esta gestando, además presentamos el significado que tienen las palabras que conforman el título de esta tesis y a continuación la evolución, clasificación, sus características y las tendencias que influirán en el ámbito del desarrollo de software.

En el cuarto capítulo *Marco metodológico* se comprueba la hipótesis que no ha sido modificada a lo largo de los estudios que se han hecho en los capítulos consecutivos al planteamiento del problema (Marco teórico y conceptual), esto se hizo con las opiniones provenientes de los profesionales en el ramo. Se aplicó un cuestionario y se analizaron las respuestas de acuerdo a los conocimientos adquiridos. Con lo cual se llegó a la comprobación de nuestra hipótesis y a la conclusión de que: efectivamente el desconocimiento de los aspectos que se deben tomar en cuenta al elegir una metodología para el desarrollo de software, hace por principio de cuentas que se hagan mal las cosas.

El quinto capítulo *Marco instrumental* es una descripción de las actividades y el plan de trabajo que emprendimos para difundir la información obtenida del tema metodologías de desarrollo de software en el área de informática, con el fin de que a las empresas o personas interesadas les sea de utilidad en la realización de sus proyectos informáticos.

1 MARCO CONTEXTUAL

1.1 Antecedentes

Hoy en día el desarrollo de software es una actividad imprescindible en múltiples entidades organizacionales, esto trae como consecuencia un amplio mercado que lucha porque el producto entregado a los clientes y/o usuarios, sea una aplicación de calidad que responda adecuadamente a sus necesidades específicas dentro del tiempo y costos requeridos.

Para ello han existido múltiples formas que han querido llegar a este cometido, como lo es la ingeniería de software. Dentro de la ingeniería de software existen modelos, habilidades técnicas y organizacionales, que toman una reflexión primaria de los diferentes enfoques para desarrollo de sistemas, en conjunto proporcionan una sana base que sustenta las metodologías ya existentes (Yourdon, RUP, programación extrema entre otras.), además dan la pauta para la creación de otras nuevas, con la combinación tanto de las metodologías como de los elementos que les dieron origen.

Los clientes demandan sistemas informáticos que cubran sus necesidades individuales, para esto es indispensable que el equipo encargado de llevar acabo el proyecto utilice los métodos necesarios para la creación del sistema informático de una manera eficaz y eficiente, que conlleve a un producto que satisfaga al cliente, cubriéndose así el ciclo de desarrollo de software. Existen muchas y muy variadas metodologías que se adaptan mejor a las posibilidades de crear un sistema informático, que pueda cubrir los requerimientos por los que se esta llevando acabo este proyecto. Muchas veces se ignoran todos estos enfoques y se procede al código cuando sólo una parte del diseño total se ha concebido. Las presiones comerciales a corto plazo a menudo hacen que esto sea casi inevitable.

El conocimiento y aplicación de cada una de estas metodologías de forma realista y consciente, trae como consecuencia nuevas alternativas de hacerlo todo de la mejor manera posible, y esto se verá reflejado en los resultados que serán, un software de calidad y lo más importante, los errores en diseño y código son descubiertos y rectificadas en la oportunidad más temprana.

1.2 Identificación del problema

Existe en este universo de creación de sistemas informáticos, algunas situaciones donde no se obtiene un producto de calidad; es decir, los tiempos y resultados del desarrollo del sistema o proyecto informático no coinciden en la mayoría de las veces con lo estipulado, lo que crea inconformidad e insatisfacción del cliente. Este tipo de problemática se da tanto en empresas consultoras como en áreas funcionales específicas donde el desarrollo de software es una actividad primordial.

Las metodologías de desarrollo de software tienen como función principal dar la forma de cómo hacer las cosas en el ámbito de desarrollo de sistemas informáticos, con lo cual podremos elegir de entre cada una de ellas la óptima, para cubrir nuestras necesidades específicas.

1.3 Demarcación del fenómeno

Esta problemática ocurre en algunas organizaciones o áreas relacionadas con el desarrollo de sistemas, que proporcionan servicios a diversos clientes mediante la creación de sistemas o proyectos informáticos empleando una metodología.

1.4 Conocimiento empírico en el medio

Se aplicó un cuestionario a personas calificadas con conocimientos teóricos sobre el tema con el fin de conocer sus opiniones.

1.4.1 Nivel de las personas entrevistadas

En este caso se eligieron a personas con conocimientos teóricos no necesariamente prácticos, que han obtenido durante la licenciatura en informática.

1.4.2 Objetivo del cuestionario

Se pretende averiguar que papel tienen las metodologías de desarrollo de software en la actualidad en universidades y empresas relacionadas a esta actividad. Ver Cuestionario para obtener conocimiento empírico en anexo A.

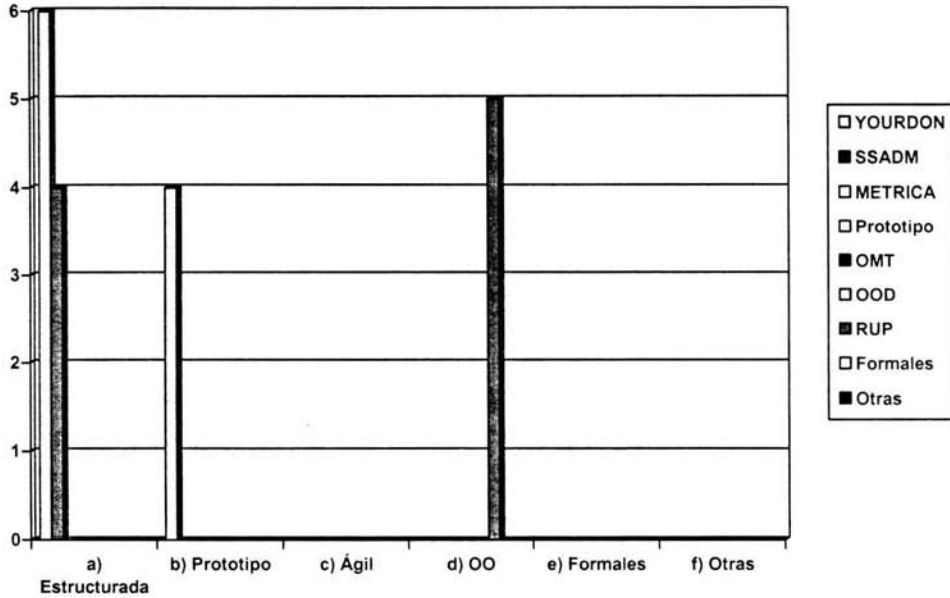
1.4.3 Personas entrevistadas

Nombre	Puesto	Nivel de estudios	Organización
Esteban Aguilar Landa	Auxiliar del área de sistemas	Pasante de Lic. en Informática	Oficina del Auditor Interno de la UNAM
José Alfredo Escorza Escorza	Analista de calidad	Lic. en Informática	Adam Technologies S. A. de C. V.
Erick Alvarado Ramírez	Desarrollador	Lic. en Informática	Gigante S. A. de C. V.
Karla Lorena Pérez Islas	Analista de sistemas	Pasante de Lic. en Informática	Subdirección de Sistemas de DGSCA-UNAM
Guillermo Reyes Cordero	Auxiliar del área de sistemas	Lic. en Informática	BANCOMER
Isabel Vilchis González	Consultor de soporte técnico	Lic. en Informática	Grupo Sicoss

1.4.4 Conclusión de cada pregunta

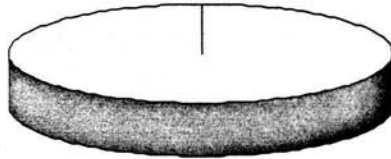
Para consultar la recopilación correspondiente a los cuestionarios de esta sección. Ver en anexo B.

Pregunta 1. ¿Qué metodologías de desarrollo de software conoce?



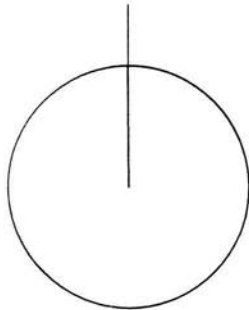
La mayoría de los entrevistados opina que las metodologías de desarrollo de software más utilizadas en la actualidad son las metodologías estructuradas como Yourdon, así como las metodologías de prototipo y las metodologías orientadas a objetos como RUP.

Pregunta 2: ¿Cree usted que llevar acabo el desarrollo de software mediante una metodología impone un proceso disciplinado?



- a) Sí
- b) No
- c) No sé

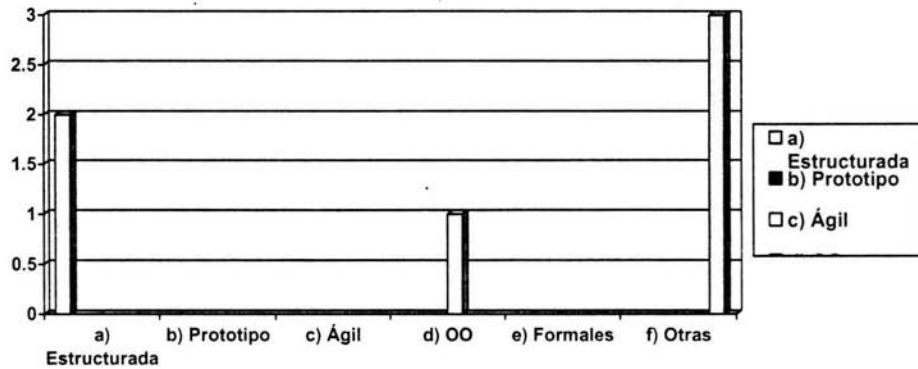
Pregunta 3: ¿Considera que una metodología hace que el desarrollo de software sea más predecible y eficiente?



- a) Sí
- b) No
- b) No

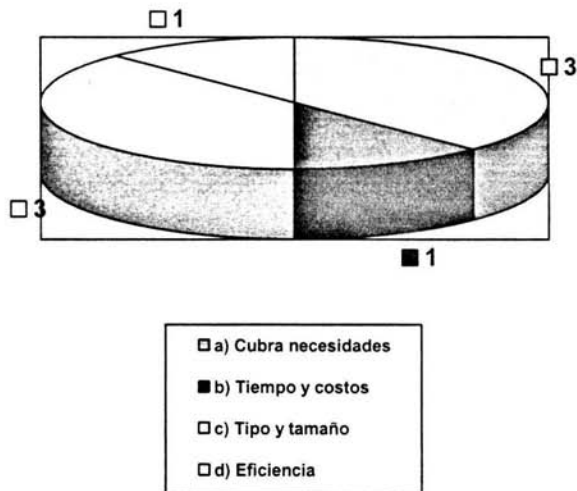
También se comprobó a través de estas personas que una metodología implica una disciplina en el desarrollo de software y además lo hace más predecible y eficiente.

Pregunta 4: ¿Cuál cree que sea la mejor metodología de desarrollo de software?

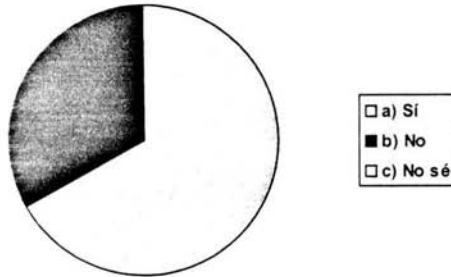


La mayor parte de los entrevistados coinciden que dependiendo del sistema que se va a desarrollar, se puede catalogar a una metodología como la mejor y otros defienden a la metodología estructurada como la más óptima.

Pregunta 5: ¿Cuál considera que sea el factor más importante para la elección de una metodología de desarrollo de software?

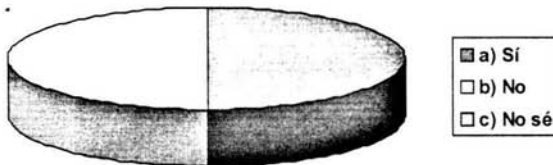


Pregunta 6: ¿Cree usted que existe una metodología de desarrollo de software eficaz para cada tipo de proyectos?

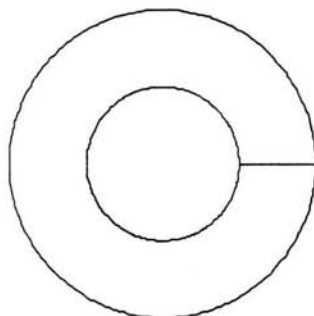


Que cubra las necesidades específicas de cada proyecto y el tipo y tamaño del proyecto son los factores determinantes para la elección de una metodología. La mayoría de los entrevistados coinciden que determinado proyecto involucra una metodología que se integre a sus necesidades específicas.

Pregunta 7: ¿Considera que se puede prescindir del análisis detallado en proyectos pequeños?



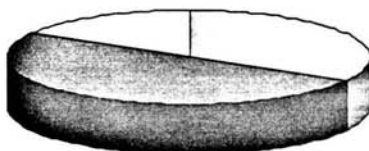
Pregunta 8: ¿Cree usted que es necesario el análisis detallado en proyectos grandes o de gran envergadura?



- a) Sí
- b) No
- c) No sé

Algunos entrevistados opinan que se podría dejar de lado el análisis profundo en proyectos pequeños pero de ninguna manera un proyecto grande podría sobrevivir sin él.

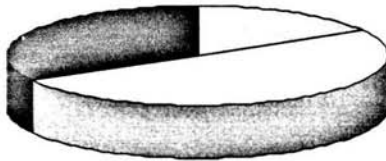
Pregunta 9: ¿Qué opina acerca del desenvolvimiento de las metodologías estructuradas?



- a) Excelente
- b) Bueno
- c) Regular
- d) Malo
- e) No sé

Los entrevistados coinciden acerca de que las metodologías estructuradas tienen una gran importancia en este ámbito.

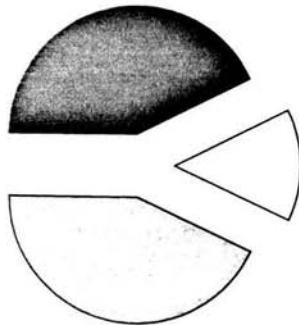
Pregunta 10: ¿Qué opina acerca del desenvolvimiento de las metodologías orientadas a objetos?



- a) Excelente
- b) Bueno
- c) Regular
- d) Malo
- e) No sé

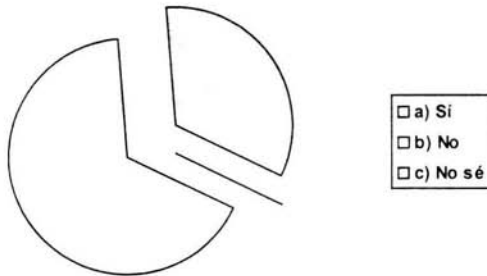
La metodología orientada de objetos esta en boca de todos pero no se sabe de alguien que haya trabajado con ellas.

Pregunta 11: ¿Cree usted que las metodologías estructuradas retarden el proceso de desarrollo?



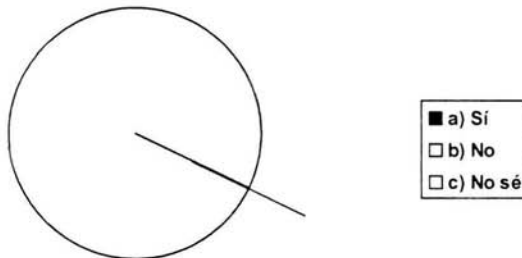
- a) Sí
- b) No
- c) No sé

Pregunta 12: ¿Cree usted que las metodologías orientadas a objetos retarden el proceso de desarrollo?

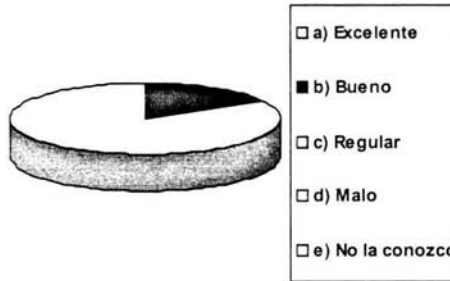


Casi todos los entrevistados opinan que una metodología estructurada no retarda el proceso de desarrollo ni mucho menos, pero otros opinan que se debe ser más flexible a la hora de exigir documentación y análisis que después no será de gran utilidad. Lo mismo sucede con las metodologías orientadas a objetos.

Pregunta 13: ¿Conoce algún proyecto de software que haya fracasado por causa de una metodología de desarrollo de software?



Pregunta 14: ¿Qué opina acerca del desenvolvimiento de las metodologías ágiles?



Todos los entrevistados no han visto fracasar a ningún proyecto por una mala metodología, más bien creen que ha sido un mal análisis o el no llevar una. Casi todos los entrevistados no conocen las llamadas metodologías ágiles de desarrollo como lo es la programación extrema.

1.5 Opiniones profesionales

Se aplicó un cuestionario a personas calificadas profesionalmente sobre el tema, con el fin de conocer sus opiniones.

1.5.1 Nivel de las personas entrevistadas

Se eligieron a personas con experiencia profesional en el tema, ya sea desarrolladores que apliquen todos sus conocimientos, líderes de proyecto o jefes del área de sistemas.

1.5.2 Objetivo del cuestionario

Se pretende llevar a cabo una averiguación acerca de las metodologías de desarrollo de software que se aplican en la actualidad en empresas y áreas relacionadas donde el desarrollo de sistemas es la actividad primordial. Ver cuestionario para obtener opiniones profesionales en anexo C.

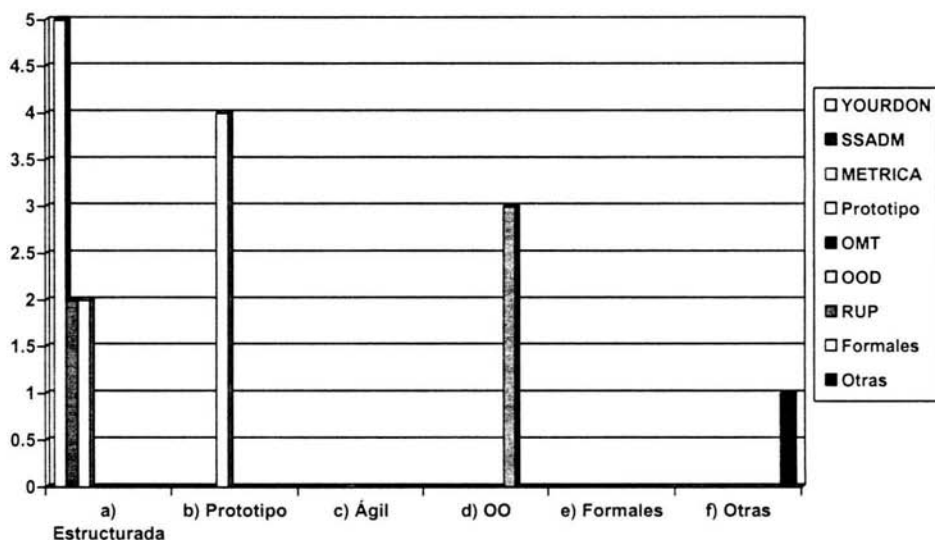
1.5.3 Personas entrevistadas

Nombre	Puesto	Organización
Karla Arias Mondragón	Líder de proyectos	Grupo Mexicano de Seguros.
Luis Carlos Infante Cisneros	Analista de sistemas	Cacto Arte e Ideas
Cecilia Herrera Marín	Administrador de proyectos	Lotería Nacional para la Asistencia Pública
Anabel Martínez Colín	Consultor-programador	Comisión Nacional del SAR
Omar Eduardo Ortiz Garza	Líder de proyecto	TBANC S. A. de C. V.
Elsa Patricia Urtecho Altamirano	Consultor en sistemas	Riverland de México S. A. de C. V.

1.5.4 Conclusión de cada pregunta

Para consultar la recopilación correspondiente a los cuestionarios de esta sección. Ver en anexo D.

Pregunta 1. ¿Qué metodologías para el desarrollo de software conoce? Escriba el inciso y las metodologías que conoce del mismo.

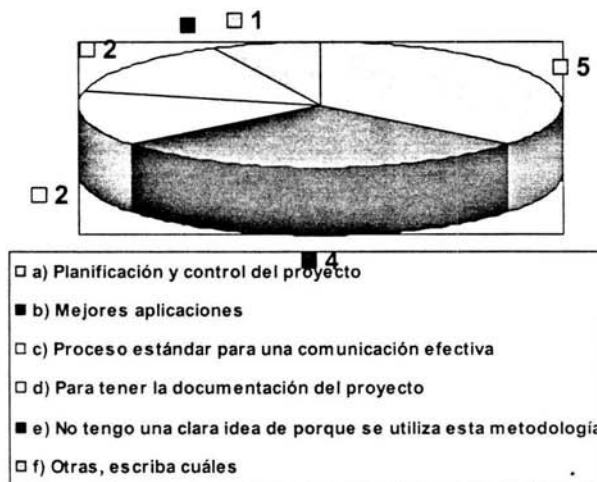


La mayoría de los entrevistados opina que las metodologías de desarrollo de software más utilizadas en la actualidad son las metodologías estructuradas como Yourdon, así como las metodologías de prototipo y las metodologías orientadas a objetos como RUP.

Pregunta 2: ¿Utiliza alguna metodología de desarrollo de software?, ¿Cuál es?

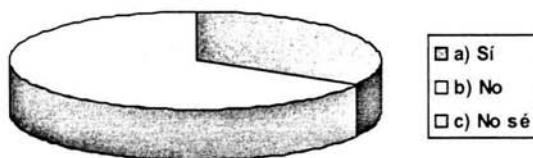
Se deduce que la mayoría de personas no utilizan alguna metodología en específico, si no que utilizan parte de alguna y/o las combinan para ajustarla al tipo de desarrollo necesario. Lo que arrojan los resultados es que utilizan: Yourdon, RUP, estructurada propia de la empresa, una combinación entre la Clásica y la Métrica 3; no, usamos herramientas de diversas metodologías; no estoy utilizando actualmente una metodología.

Pregunta 3: ¿Específicamente para qué usa esta metodología para el desarrollo de software?



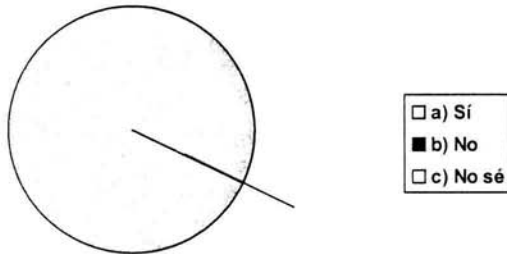
La mayor parte de los entrevistados coincide en que utiliza alguna de las metodologías de desarrollo para la planificación y control del proyecto así como para obtener mejores aplicaciones, también mencionan que la usan para crear una cultura en la empresa.

Pregunta 4: ¿Aplica formalmente todos los elementos de la notación de esta metodología?



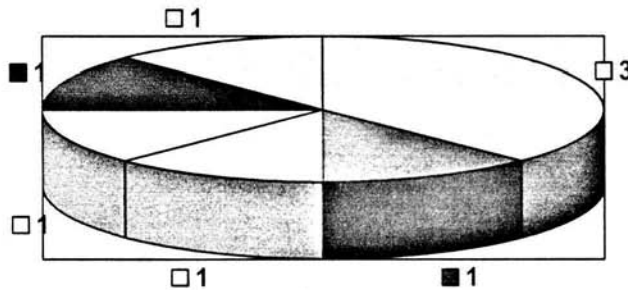
La mayor parte de los entrevistados dice que no aplica formalmente los elementos de la notación de una metodología, porque depende de varios factores como son: no aplican al proyecto, hacen combinación de varias metodologías, depende de la formación de los colaboradores y de las políticas de la empresa.

Pregunta 5: ¿Considera que se han obtenido beneficios en el desarrollo de sistemas a partir del uso de esta metodología?



La mayor parte de los entrevistados dicen que si consideran que se han obtenido beneficios con el uso de una metodología. Pues permiten diferentes factores: un mejor control y organización del sistema en desarrollo, un orden y se cumplen y/o exigen los entregables, reducción en tiempo de desarrollo en fallas y en mantenimiento.

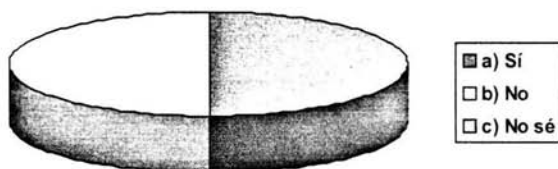
Pregunta 6: ¿Cuál considera que es la razón por la que han obtenido o no beneficios?



- a) Planificación y control del proyecto
- b) Existencia de reglas predefinidas
- c) El uso de la documentación para lograr una comunicación efectiva
- d) Especificar claramente los requerimientos, no permitir flexibilidad en los cambios
- e) No se ha obtenido ningún beneficio
- f) Otros, escriba cuales son:

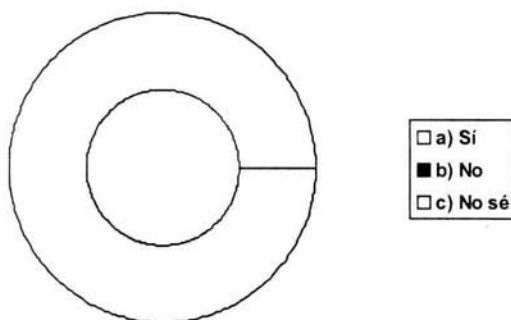
Los entrevistados dicen que se han obtenido beneficios como son la planificación y control, existencias de reglas predefinidas, uso de documentación y especificación clara de requerimientos así como por el hecho de realizar un buen análisis y desarrollo del software.

Pregunta 7: ¿Ha sido complicado incorporar esta metodología?



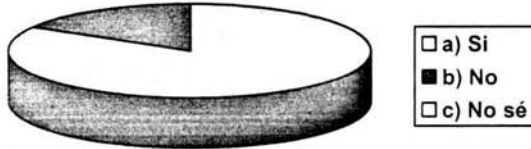
Algunos entrevistados opinan que es complicado incorporar la metodología, porque es difícil para los colaboradores seguir las reglas, porque la alta dirección no apoya esto y algunas veces por falta de tiempo.

Pregunta 8: ¿Considera que el uso de esta metodología ha facilitado el desarrollo de sistemas?



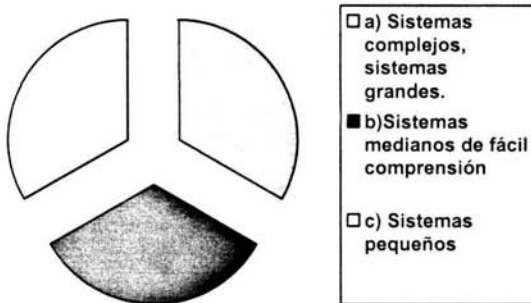
Los entrevistados acuerdan que las metodologías facilitan el desarrollo de software ya que todos hablan el mismo idioma y se maneja mejor la información.

Pregunta 9: ¿Es conocida y comprendida la metodología de desarrollo de software que utiliza en su organización?



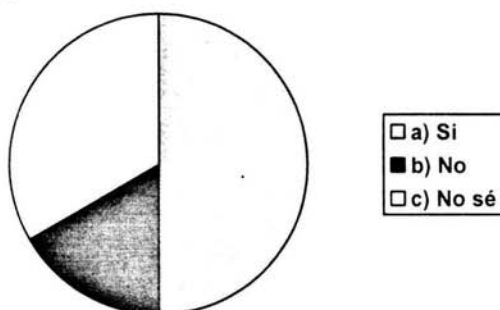
Los entrevistados dicen que es conocida y comprendida la metodología de desarrollo en sus organizaciones.

Pregunta 10: ¿Qué tipo de proyectos desarrolla con esta metodología de desarrollo de software?



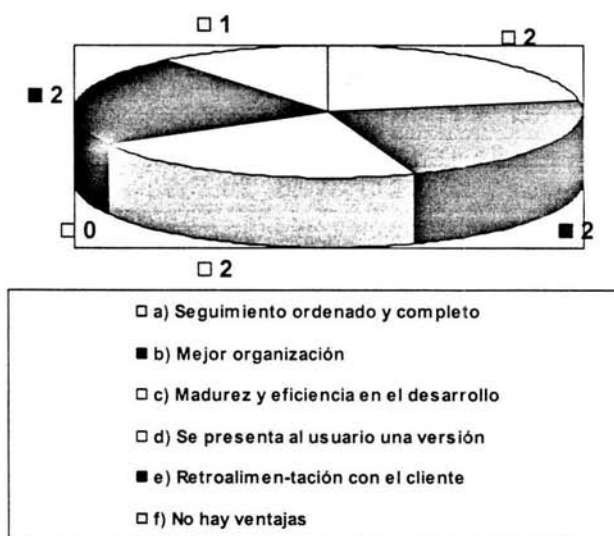
La mayoría de los entrevistados dicen que desarrollan proyectos grandes, medianos y pequeños en sus organizaciones.

Pregunta 11: ¿Cree usted que la metodología que utiliza es la mejor?



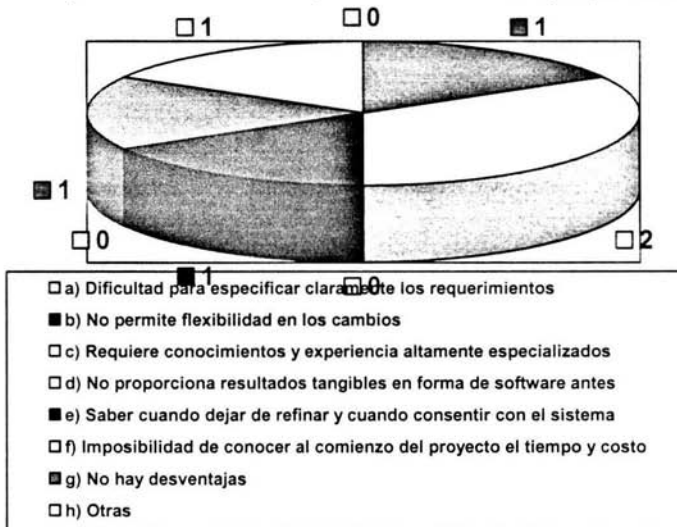
Varios entrevistados opinan que la metodología que utilizan es la mejor.

Pregunta 12: ¿Cuáles son las ventajas de usar la metodología que usted utiliza?



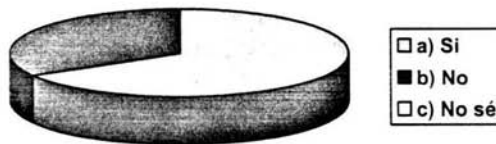
Los entrevistados opinan que las ventajas de la metodología que utilizan son: seguimiento ordenado y completo, mejor organización, madurez y eficiencia en el desarrollo y retroalimentación con el cliente.

Pregunta 13: ¿Cuáles son las desventajas de usar la metodología que usted utiliza?



Los entrevistados opinan que las desventajas de la metodología que utilizan son principalmente: que requiere conocimientos y experiencia altamente especializados, que no permite flexibilidad en los cambios para saber cuando dejar de refinar y cuando consentir con el sistema.

Pregunta 14: ¿Conoce algún proyecto de desarrollo de sistemas que haya fracasado por causa de una metodología de desarrollo software?



La mayoría de los entrevistados no han visto fracasar a ningún proyecto por una mala metodología, más bien creen que ha sido el hecho de que no se implemento de manera adecuada.

1.6 Conclusión general

Es de gran utilidad la información obtenida de las personas que tienen conocimiento teórico y/o práctico sobre las metodologías de desarrollo de sistemas informáticos, ya que gracias a sus respuestas nos dan un amplio panorama para el desarrollo de la presente investigación, de acuerdo con lo anterior identificamos que las metodologías de desarrollo de software más utilizadas en la actualidad son las metodologías estructuradas como Yourdon, SSADM; así como las metodologías de prototipo y las metodologías orientadas a objetos como RUP.

Identificamos que efectivamente las metodologías de desarrollo de software imponen rigor, detalle y precisión en el desarrollo de software con lo cual se puede crear un producto de calidad y tener satisfecho al cliente. Así como observamos que cada tipo de proyecto exige una metodología eficaz y eficiente, y además, que cubra sus necesidades específicas es por ello que la información recopilada nos sirve de base para establecer una clasificación que nos guíe para identificar la metodología óptima para cada tipo de proyecto.

1.7 Hipótesis preliminar

Causas	Efectos
Carencia de conocimientos sobre aspectos importantes al elegir una metodología de desarrollo de software, que cubra las necesidades específicas del proyecto a desarrollar.	Un desarrollo de software impredecible e ineficiente, los tiempos y resultados del desarrollo no coinciden con lo estipulado. Inconformidad e insatisfacción del cliente

Planteamiento de la hipótesis: “La carencia de conocimientos sobre aspectos importantes al elegir una metodología de desarrollo de software, que cubra las necesidades específicas del proyecto a desarrollar, origina un desarrollo de software carente de grado de predicibilidad y eficiencia, los tiempos y resultados del desarrollo no coinciden con lo estipulado, generando así inconformidad e insatisfacción del cliente”.

1.8 Objetivos

Personales:

- Culminar con nuestros estudios profesionales.
- Aumentar el conocimiento sobre el desarrollo de sistemas y sus metodologías.
- Fomentar el interés en este tema.

Particulares:

- Obtener el Título de Licenciado en Informática con el cumplimiento de los requisitos que establece la Universidad Nacional Autónoma de México en el Reglamento General de Exámenes. Capítulo IV (Exámenes Profesionales y de Grado, Artículos 18, 19, 20 y 21), así como el Reglamento de Exámenes Profesionales de la Facultad de Contaduría y Administración (Título I, artículos 3, 7, 9; Título II inciso E, artículos 44-56).

Generales:

- Establecer un documento en el cual las personas interesadas puedan encontrar información sobre las metodologías de desarrollo de software.
- Se pretende crear un documento que sienta las bases para clasificar las metodologías de acuerdo a las necesidades específicas de cada proyecto y así saber cuál es la idónea para llevar a cabo dicho proyecto.

2 MARCO TEÓRICO

2.1 Acopio de libros

Tipo de lectura:

De estudio: Lectura profunda de toda o casi toda a obra.

Ligera: Lectura de algún capítulo.

Rápida: Lectura de algún tema.

2.1.1 Libros de estudio

Título	Ingeniería de Software
Autor	Roger S. Pressman
Editorial	Mc. Graw Hill
Edición	España 1998
ISBN	84-481-1186-9
Comentario	
<p>PARTE I . EL PRODUCTO Y EL PROCESO</p> <p>Capítulo 1. El producto.</p> <p>Este capítulo trata sobre el software, que es un elemento clave en la realización de esta tesis ya que es propiamente del desarrollo del software al cual va enfocada.</p> <p>Estudia el origen del software y menciona que el contexto en el que se ha desarrollado el software está fuertemente ligado a las cuatro décadas de evolución de los sistemas informáticos, además cita las características del software, los componentes que tiene y las aplicaciones que puede tener.</p> <p>No siendo de menor importancia el que otorgue conceptos sobre crisis del software e ingeniería del software que son clave en el origen y los fundamentos de las metodologías de desarrollo de software, que es el tema principal de este trabajo.</p> <p>Capítulo 2. El proceso.</p> <p>El énfasis de este capítulo recae en el proceso, de esta manera define lo que es un proceso de software, los métodos y las herramientas que pueden ayudar a poblar el proceso de desarrollo de software. Menciona que la ingeniería de software es el análisis, diseño, construcción, verificación y gestión de entidades técnicas o sociales. El trabajo que se asocia a la ingeniería del software se puede dividir en tres fases</p>	

genéricas: *fase de definición, fase de desarrollo y fase de mantenimiento.*

Presenta los diferentes *modelos de ciclo de vida del software* existentes y su marcada diferenciación al estudiar las características de cada una.

PARTE II. GESTION DE PROYECTOS DE SOFTWARE

Capítulo 3. Conceptos sobre gestión de proyectos.

El tema central de este capítulo es la gestión de proyectos, por lo que nos da una introducción de lo que es. Cita que existen tres “P’s”, que tienen una influencia sustancial en la gestión de proyectos de software: *personal, problema y proceso*, propone al personal como el elemento fundamental en todos los proyectos de software.

Capítulo 4. El proceso de software y métricas del proyecto.

El contenido de este capítulo se centra en la medición que permite que gestores y profesionales mejoren el proceso de software: ayudan en la planificación, seguimiento y control de un proyecto de software; evalúan la calidad del producto. Menciona que las métricas del proceso permiten que una organización tome una visión estratégica proporcionando mayor profundidad de la efectividad de un proceso de software.

Capítulo 5. Planificación de proyectos de software

Habla sobre la planificación de un proyecto de software establece que se deben *estimar tres cosas antes de que comience el proyecto: duración, esfuerzo y personal*. El planificador debe predecir los recursos y el riesgo implicado. Enumera varias técnicas que nos ayudan a obtener estimaciones exactas para un proyecto como son: *técnicas de descomposición y técnicas empíricas*.

Capítulo 6. Gestión del riesgo

Este capítulo se centra en la gestión de riesgos. Menciona que cuando se pone mucho en juego el sentido común en un proyecto de software, aconseja realizar un análisis de riesgos, ya que proporciona menos trastornos, durante el proyecto, una mayor habilidad de seguir y controlar el proyecto y la confianza que da planificar los problemas antes de que ocurran, valiendo la pena el tiempo que se invierte en la gestión de riesgos.

Capítulo 7. Planificación temporal y seguimiento del proyecto.

Menciona que la planificación temporal es la culminación de una actividad de planificación, componente primordial de la dirección de proyectos de software. El gestor del proyecto puede seguir y controlar todos los pasos del proceso de ingeniería de software usando la planificación temporal como directriz.

Capítulo 8. Control de calidad del software

La garantía de calidad del software (SQA) es una actividad de protección que se aplica a cada paso del proceso de ingeniería del software. *La SQA combina procedimientos para la aplicación efectiva de métodos y herramientas, las revisiones técnicas formales, las técnicas y estrategias de prueba, los procedimientos de control de cambios, los procedimientos de garantías de ajuste a los estándares y los mecanismos de medida e información.*

Capítulo 9. Gestión de la configuración del software.

Trata sobre la configuración de software que está compuesta por un conjunto de objetos interrelacionados denominados elementos de configuración de software, que se producen como resultado de alguna actividad de la ingeniería del software. Menciona lo que es una línea base, control de cambios y auditoría de configuración.

PARTE III. METODOS CONVENCIONALES DE LA INGENIERIA DE SOFTWARE

Capítulo 10. Ingeniería de sistemas

En la ingeniería de sistemas se analizan el dominio de negocio o producto para establecer todos los requisitos básicos. La ingeniería de información es un enfoque de la ingeniería de sistemas de computadora que se usa para definir arquitecturas que permiten a un negocio utilizar la información eficazmente.

Capítulo 11. Conceptos y principios del análisis

Como resultado del análisis se desarrolla la especificación de requisitos de software. La revisión es esencial para asegurarse de que el cliente y el desarrollador tienen el mismo concepto del sistema.

Capítulo 12. Modelado del análisis

Este capítulo se centra en el análisis estructurado que es el método más usado para el modelado de requisitos, utiliza el modelo de datos y el modelo de flujos para crear la base de un adecuado modelo de análisis.

Capítulo 13. Conceptos y principios del diseño

Este capítulo menciona que el diseño es el núcleo técnico de la ingeniería de software. Durante el diseño se desarrollan, revisan y documentan refinamientos progresivos de estructuras de datos, arquitectura del programa, interfaces y detalles procedimentales. El diseño da como resultado representaciones del software para valorar su calidad.

Capítulo 14. Métodos de diseño

Los métodos de diseño presentados en este capítulo llevan a un modelo de diseño del software, se desarrolla la estructura de datos, se establece la arquitectura del programa, se definen los módulos y se establecen las interfaces. Este proyecto de implementación forma la base de todo el subsiguiente trabajo de ingeniería del software.

Capítulo 15. Diseño de sistemas de tiempo real.

El tema principal de este capítulo trata sobre el diseño de software de tiempo real que abarca los aspectos del diseño convencional de software, aunque, al mismo tiempo introduce un nuevo conjunto de criterios y aspectos de diseño. Debido a que el software de tiempo real debe responder a sucesos del mundo real en un tiempo dictado por dichos sucesos, cualquier tipo de diseño se vuelve más complejo.

Capítulo 16. Métodos de prueba del software

Este capítulo menciona que cada vez que el cliente usa el programa lleva a cabo una prueba. Aplicando el diseño de casos de prueba, el ingeniero del software puede conseguir una prueba más completa y descubrir y corregir así el mayor número de errores antes de que comiencen las pruebas del cliente.

Capítulo 17. Estrategias de prueba del software

El contenido de este capítulo gira en torno a la prueba del software que contabiliza el mayor porcentaje del esfuerzo técnico de los procesos de desarrollo de software. El objetivo de la prueba del software es descubrir errores, para conseguir este objetivo, se planifica y se ejecutan una serie de pasos:

- Pruebas de unidad
- Pruebas de integración
- Pruebas de validación
- Pruebas del sistema

Capítulo 18. Métricas técnicas del software

Este capítulo trata de las métricas del software que proporcionan una manera cuantitativa de valorar la calidad de los atributos internos del producto, permitiendo

por tanto al ingeniero valorar la calidad antes de construir el producto.

PARTE IV. INGENIERÍA DE SOFTWARE ORIENTADA A OBJETOS

Capítulo 19. Conceptos y principios orientados a objetos

Este capítulo se centra en las tecnologías de objetos que reflejan una visión natural del mundo. Los productos y sistemas orientados a objetos son producidos usando un modelo evolutivo, algunas veces llamado recursivo/paralelo. El software orientado a objetos evoluciona iterativamente y debe dirigirse teniendo en cuenta que el producto final se desarrollara a partir de una serie de incrementos.

Capítulo 20. Análisis orientado a objetos

El contenido de este capítulo abarca los métodos de análisis orientado a objetos que permiten al ingeniero del software modelar un problema a través de la representación de objetos, atributos, y operaciones como componentes primarias del modelado.

Capítulo 21. Diseño orientado a objetos

Este capítulo se centra en el diseño orientado a objetos. Este proceso se puede describir como una pirámide compuesta de cuatro capas, *la capa base* se centra en el diseño de los subsistemas que implementan las funciones del sistema más importantes; *la capa de las clases* especifica la arquitectura de objetos general y la jerarquía de clases necesaria para implementar el sistema; *la capa de mensajes* indica como desarrollar la colaboración entre objetos; y *la capa de responsabilidades* identifica las operaciones y atributos que caracterizan cada clase.

Capítulo 22. Pruebas orientadas a objetos

Este capítulo trata del objetivo general de las pruebas orientadas a objetos que es encontrar el número de errores máximo con el mínimo esfuerzo.

Capítulo 23. Métricas técnicas para sistemas orientadas a objetos

Este capítulo analiza las métricas para los sistemas orientados a objetos, que se centran en métricas que se pueden aplicar a las características de diseño y de clase: localización, encapsulamiento, ocultamiento de información, herencia y técnicas de abstracción de objetos; que hagan única a esta clase.

PARTE V. TEMAS AVANZADOS DE INGENIERIA DE SOFTWARE

Capítulo 24. Métodos formales

Este capítulo trata principalmente de los métodos formales; estos métodos ofrecen un fundamento para entornos de especificación que dan lugar a modelos de análisis más completos, consistentes y carentes de ambigüedad que aquellos que se producen empleando métodos convencionales u orientados a objetos.

Capítulo 25. Ingeniería del software de sala limpia

La ingeniería del software de sala limpia es un enfoque formal para el desarrollo del software, que pueda dar lugar a un software que posea una calidad notablemente alta.

Capítulo 26. Reutilización del software

Este capítulo se centra en la reutilización del software, menciona las técnicas de análisis y diseño de componentes reutilizables que se basan en los mismos principios y conceptos que forman parte de las buenas prácticas de ingeniería del software.

Capítulo 27. Reingeniería

El contenido de este capítulo se centra alrededor de la reingeniería del software que abarca una serie de actividades de entre las que se incluye, el análisis de inventario, la reestructuración de documentos, la ingeniería inversa, la reestructuración de programas y datos, y la ingeniería progresiva.

Capítulo 28. Ingeniería del software cliente/servidor

Este capítulo trata sobre la personalización del enfoque de ingeniería de software que se debe dar a los sistemas cliente/servidor, debido a sus características de arquitectura especiales.

Capítulo 29. Ingeniería de software asistida por computadora

El tema principal de este capítulo son las herramientas de ingeniería de software que abarcan todos los pasos del proceso de software, y también todas aquellas actividades generales que se aplican a lo largo de todo el proceso.

Capítulo 30. El futuro

En este capítulo se examinan las tendencias futuras, en donde se intentará explorar el ámbito del cambio y la forma en que el cambio en sí va a afectar al proceso del software.

Título	Análisis y Diseño Orientado a Objetos
--------	---------------------------------------

Autor	Grady Booch
Editorial	Addison Wesley / Díaz de Santos
Edición	Segunda edición 1998
ISBN	0-201-60122-2
Colocación Biblioteca	QA76.64 B6618

Comentario

PRIMERA SECCION: CONCEPTOS

Capítulo 1. Complejidad

Este capítulo describe a profundidad la complejidad que puede tener el software, derivándose de cuatro elementos:

- Complejidad del dominio del problema.
- Dificultad de gestionar el proceso de desarrollo.
- Flexibilidad que se puede alcanzar a través del software.
- Problemas que plantea la caracterización del comportamiento de sistemas discretos.

Presenta ejemplos de sistemas complejos tratando de definir su estructura y estableciendo cinco atributos comunes a todos los sistemas complejos. Sugiere formas de cómo imponer orden al caos, es decir, dominar la complejidad del sistema con técnicas como: *la descomposición (la cual enumera varios tipos), la abstracción y la jerarquía.*

Incluye un recuadro en donde se especifica de manera resumida las categorías de métodos de diseño, donde presentan los términos; método y metodología. Por último menciona de diseño de sistemas complejos, especificando su significado y la importancia de construir un modelo, así como, los elementos de diseño de software y los modelos de desarrollo orientado a objetos.

Capítulo 2. El modelo de objetos.

En este capítulo se muestra claramente que es y que no es el diseño orientado a objetos, y en que se diferencia de otros métodos a través del uso de los siete modelos de objetos, destacando que la tecnología orientada a objetos, se apoya en los sólidos fundamentos de la energía y que abarca los principios de abstracción, encapsulación, modularidad, jerarquía, tipos, concurrencia y persistencia. El diseño orientado a objetos es fundamentalmente diferente a los enfoques de diseño estructurado tradicionales ya que requiere distinto modo de pensar y la arquitectura software que

produce.

Capítulo 3. Clases y objetos

Este capítulo se vuelca en un estudio detallado de la naturaleza de las clases, los objetos y sus relaciones, también ofrecen varias reglas para construir abstracciones y mecanismos de calidad, tomando muy en cuenta que cuando se utilizan metodologías orientadas a objetos para analizar o diseñar un sistema de software complejo, los bloques básicos de construcción son las clases y objetos.

Capítulo 4. Clasificación

Este capítulo se centra principalmente en la problemática que se presenta a la hora de identificar clases y objetos en el diseño orientado a objetos, ya que implica descubrimiento e invención, menciona que la clasificación es un proceso incremental e iterativo, que se complica por el hecho de que un conjunto de dos objetos puede clasificarse de muchas formas igualmente correctas. Tres enfoques de la clasificación:

- Categorización clásica (clasificación por propiedades).
- Agrupamiento conceptual (clasificación por conceptos).
- Teoría de prototipos (clasificación por asociación de un prototipo).

SEGUNDA SECCION: EL METODO

Capítulo 5. La notación

En el diseño de un sistema más complejo es importante verlo desde muchas perspectivas a saber, su estructura física y lógica, y su semántica estática y dinámica. La notación para el desarrollo orientado a objetos incluye cuatro diagramas básicos:

- Diagramas de clases
- Diagramas de objetos
- Diagramas de módulos
- Diagramas de procesos
- Diagramas de transición de estados (suplementario)
- Diagramas de intersección (suplementario)

Capítulo 6. El proceso

En este capítulo se examina el proceso de diseño orientado a objetos incremental e iterativo en detalle, y se considera el propósito, productos, actividades y medidas de sus diversas fases. El proceso del diseño orientado a objetos no se puede describir mediante recetas es decir: la documentación nunca debería dirigir el proceso de desarrollo, esta lo bastante bien definido como para ofrecer un proceso predecible y repetible para la organización de desarrollo de software.

Título	Desarrollo y Gestión de Proyectos Informáticos
Autor	Steve McConell
Editorial	Mc. Graw Hill
Edición	1996 España
ISBN	84-481-1229-6

Comentario

PARTE I. DESARROLLO EFICIENTE

Capítulo 1. Bienvenido al desarrollo rápido

Este capítulo tiene una breve explicación acerca del desarrollo rápido, que significa desarrollar software a una velocidad superior a la alcanzada en este momento. Hace un especial énfasis en que la única prioridad real en el desarrollo de un proyecto es entregarlo lo antes posible, que es propio de este tipo de desarrollo. Además menciona que la velocidad con que desarrollemos cualquier programa concreto dependerá de la proporción de métodos eficaces y de métodos orientados que seleccionemos, para así lograr la velocidad en el desarrollo.

Dentro de los métodos orientados a la planificación están los métodos que permiten desarrollar más rápido, los que reducen el riesgo en la planificación y los que hacen visible el progreso.

Capítulo 2. Estrategia para el desarrollo rápido

Este capítulo se centra en la estrategia general para el desarrollo rápido en la que el mejor plan posible depende de evitar los errores clásicos, las bases del desarrollo y la gestión de riesgos, además del uso de métodos orientados a la planificación. También cita cuatro dimensiones para maximizar la velocidad de desarrollo: personas, proceso, producto y tecnología. Conceptualiza diversos tipos de desarrollo rápido dependiendo de las necesidades que se tienen.

Diferentes proyectos tienen diferentes necesidades, pero en todos los casos la clave es aceptar los límites en las dimensiones que no podemos cambiar y centrarse en las otras dimensiones para obtener el resto de los beneficios que necesitamos en la planificación. Se describen además las ventajas y desventajas de los métodos llamados "métodos de codificación a destajo".

Capítulo 3. Errores clásicos

Este capítulo es dedicado al estudio de los errores clásicos en el desarrollo de software, menciona y da ejemplos de los más comunes, así como que hacer para evitarlos.

Enfatiza que para conseguir el desarrollo rápido se tiene que evitar cometer algún error muy grande, ya que con esto se puede estropear todo. Aparecen también algunos de los errores clásicos relacionados con las dimensiones de la velocidad de desarrollo: personas, proceso, producto y tecnología.

Menciona que para evitar algunos errores es necesario hacer su propia lista de malos hábitos y tratar de no cometerlos; así se aprenderá de los errores.

Capítulo 4. Bases de desarrollo de software

El capítulo menciona que para obtener éxito en el software hay que prestar la mayor atención posible a los fundamentos. Los fundamentos de gestión consisten en determinar todas las características del producto, asignar los recursos apropiados para ese producto, crear un plan para aplicar esos recursos y luego controlar y dirigir esos recursos para evitar que el proyecto se desvíe. Las bases técnicas pueden ayudar a ahorrar una gran cantidad de tiempo si se llevan al pie de la letra.

La gestión de requerimientos es simplemente vital para el desarrollo y hacerlo de acuerdo a las bases, otorgaría el éxito a dicho proyecto. Menciona lo importante que es tener todas las bases del diseño. En el momento en que se llega a la construcción ya se ha llevado acabo la mayor parte del trabajo que constituye una base sólida para el éxito o fracaso del proyecto. Así también las bases de control de calidad proporcionan un apoyo fundamental para obtener la máxima velocidad del desarrollo.

Capítulo 5. Gestión de riesgos

El capítulo contiene un excelente tutorial sobre el manejo de los riesgos, su estudio se centra en los riesgos en la planificación (ignorando los riesgos sobre costes y producción). La función de la gestión de riesgos es identificar, estudiar y eliminar las fuentes de riesgo antes de que empiecen a amenazar la finalización satisfactoria de un proyecto de software.

Se divide en dos subcategorías:

- Estimación de los riesgos: En la cual se va a identificar y medir el impacto de cada uno de esos riesgos
- Control de los riesgos: Tiene más que ver con la planificación de la gestión de los riesgos y planes para resolver cada riesgo puestos en marcha.

PARTE II. DESARROLLO RÁPIDO

Capítulo 6. Cuestiones fundamentales para el desarrollo rápido

Uno de los enfoques de la estimación de proyectos software mantiene que todo

proyecto tiene una fecha exacta en que debe ser terminado. Ya que se ha aprendido a evitar los errores clásicos y se han asimilado los fundamentos del desarrollo del software y la gestión de riesgos, se tiene las bases para centrarse en los métodos de desarrollo orientados a la planificación.

A lo largo de este capítulo también se estudiarán otras cuestiones fundamentales para el desarrollo rápido como es la posibilidad de terminar a tiempo, el que los clientes perciban los adelantos del proyecto, como se debe distribuir el tiempo empleado, establecer un equilibrio entre los factores de la velocidad de desarrollo. Los diferentes tipos de software requieren diferentes tipos de soluciones (métodos que se consideran rápidos y vastos para el desarrollo de un sistema pueden resultar excesivamente rigurosos para otros).

Lo primero que se necesita es determinar cuál es el tipo de desarrollo rápido que se necesita:

- Ligero aumento de velocidad.
- Mayor predicibilidad.
- Mejor visibilidad del progreso.
- Menores costes.
- Más velocidad a cualquier coste.

Capítulo 7. Planificación del ciclo de vida

El estudio de este capítulo se centra en el ciclo de vida de software y menciona que consiste en realizar todas las actividades comprendidas entre el momento en que se inicia la primera versión de un sistema hasta el momento en que la última versión finaliza.

De este modo analiza los siguientes modelos de desarrollo de software:

- Cascada pura
- Codificar y corregir
- Espiral
- Cascadas modificadas
- Prototipado evolutivo
- Entrega por etapas
- Diseño por planificación
- Entrega evolutiva
- Diseño por herramientas

Cita tanto las ventajas como los inconvenientes de cada una. Además enfatiza la idea de seleccionar un ciclo de vida que funcione adecuadamente con las necesidades de nuestro proyecto.

Capítulo 8. Estimación

Este capítulo ofrece una introducción a la estimación de los proyectos de software, describe como alcanzar una buena estimación que se logra manipulando los números y creando algo razonablemente preciso. Es difícil saber si es posible construir el producto que el cliente desea en el tiempo esperado hasta que se comprende perfectamente lo que el cliente quiere.

Capítulo 9. Planificación.

El capítulo trata sobre las implicaciones que trae como consecuencia el hacer una estimación de la planificación excesivamente optimista, por lo cual se debe crear una estimación precisa y tratar que esta estimación sea aceptada para que exista una disminución de la presión de la planificación ya que esto crea inconvenientes por todo el tiempo en que se esta desarrollando.

Capítulo 10. Desarrollo orientado al cliente

Este capítulo presenta algunos métodos que pueden ser de utilidad para mantener buenas relaciones con el cliente, ya que se sostiene que *la buena comunicación con el usuario final es uno de los tres factores críticos de los proyectos de desarrollo rápido, así mismo el esforzarse en comprender las expectativas del cliente puede ahorrar muchas fricciones y trabajo extra; ya que algunas de ellas son imposibles y hay que hacer que se entienda.*

Capítulo 11. Motivación

De las cuatro áreas de desarrollo rápido (personas, proceso, producto y tecnología). Las personas ofrecen la mayor posibilidad de acortar las planificaciones del software a lo largo de una serie de proyectos, la mayoría de los estudios de productividad han encontrado que la motivación tiene mayor influencia en la productividad que cualquier otro factor. Este capítulo describe a profundidad como motivar a los desarrolladores del software para mejorar la velocidad del desarrollo. Además muestra una lista ordenada de factores motivacionales para los desarrolladores, directivos y personas en general, también exhibe algunos de los factores que destruyen la moral por el trabajo.

Capítulo 12. Equipo de trabajo

El contenido de este capítulo se centra en el concepto de equipo de trabajo estableciendo los usos de los equipos de trabajo en el software, la importancia que

tiene éste para un desarrollo rápido, se enumeran las principales causas de fallo de los equipos y menciona la configuración de equipos a largo plazo así como un resumen de directrices para grupos de trabajo.

La creación de un equipo de alto rendimiento debe tener:

- Una alta visión u objetivo compartidos.
- Sentido de identidad de equipo.
- Estructura dirigida por resultados.
- Miembros de equipo competentes.
- Compromiso.
- Confianza.
- Interdependencia.
- Comunicación efectiva.
- Autonomía.
- Enriquecimiento.
- Disfrute.

Capítulo 13. Estructura del equipo

Este capítulo aporta conocimientos sobre como conformar un equipo, ya que aunque se tenga personal formado, motivado y acostumbrado a trabajar duro, una estructura errónea del equipo puede terminar su esfuerzo en lugar de lanzarlo al éxito. Una mala estructura del equipo puede aumentar el tiempo de desarrollo, reducir la calidad, deteriorar la moral, incrementar los cambios de personal, y llevar a la cancelación del proyecto. Por ello se describen algunas consideraciones sobre la estructura del equipo así como algunos modelos del equipo que existen.

Capítulo 14. Control del conjunto de prestaciones

Este capítulo explica la forma en la que se pueden controlar los cambios y entregar el software a tiempo. Hay tres tipos generales de control del conjunto de prestaciones:

- Control al principio del proyecto - Para definir un conjunto de prestaciones consistente con los objetivos de planificación y presupuesto de su proyecto.
- Control a mitad del proyecto - Para controlar los cambios de requerimientos.
- Control al final del proyecto - Recortando prestaciones para alcanzar un objetivo de planificación o de coste.

Capítulo 15. Herramientas para aumentar la productividad

Este capítulo se centra en las herramientas para incrementar la productividad, los problemas asociados con la adopción de herramientas, la formación de las personas para su uso y la estimación del ahorro obtenida por el mismo.

El uso de herramientas no es una solución a corto plazo, porque necesitan tiempo y dinero para adquirirlas y manejarlas de forma efectiva. Los mejores proyectos no utilizan necesariamente metodologías de última generación, ni hacen un amplio uso de la automatización y de las herramientas. Se basan en *principios generales como un equipo de trabajo compacto, comunicación entre las áreas y control del proyecto. Una buena organización y gestión parecen ser un factor mucho más importante para el éxito que la tecnología.*

Capítulo 16. Recuperación de proyectos

Este capítulo describe un plan riguroso de rescate para los proyectos que tienen problemas, esto no se refiere solamente a que su planificación se ha retrasado un poco, es un proyecto que esta pidiendo ayuda. Por ello menciona las características de estos proyectos y también dice que para salvar un proyecto que se encuentra en problemas, unas pequeñas modificaciones no bastaran, se necesita tomar una fuerte acción de corrección.

PARTE III. MÉTODOS RECOMENDABLES

Los capítulos de esta parte del libro describen los métodos recomendables para desarrollo rápido.

2.1.2 Libros de lectura ligera

Título	Análisis y Diseño de Sistemas de Información
Autor	James A. Senn
Editorial	Mc. Graw Hill
Edición	México 1998
ISBN	968-422-165-7
Colocación Biblioteca	QA76.958854418
Comentario	
Capítulo 1. Introducción al desarrollo de sistemas de información	
En este capítulo se analiza la forma de operar en el desarrollo de sistemas de información, antes de que cualquier proyecto pueda realizarse debe elaborarse un estudio del sistema, para conocer los detalles en relación con la situación actual del negocio.	
Menciona el ciclo de vida del desarrollo de sistemas que es un conjunto de actividades	

que los analistas y diseñadores llevan a cabo para desarrollar y poner en marcha el sistema de información; incluye la investigación preliminar, la recolección de datos y la determinación de requerimientos, el desarrollo del prototipo, el diseño del sistema, el desarrollo de software, la prueba de los sistemas y la puesta en marcha. El análisis y diseño de los sistemas se refiere al proceso de examinar una situación de la empresa con la intención de mejorarla mediante nuevos procedimientos y métodos.

Menciona las tareas del analista de sistemas, características de los usuarios y los diferentes tipos de sistemas.

Capítulo 2. Inicio del proyecto.

En este capítulo se muestra el origen de las solicitudes del proyecto: *gerentes de departamento, ejecutivos de alto nivel, analistas de sistemas, grupos externos* y se enumeran las razones para el inicio de un proyecto:

- Mayor velocidad en el proceso.
- Mayor exactitud y mejor consistencia.
- Consulta más rápida de la información.
- Integración de las áreas del negocio.
- Reducción de costos.
- Mayor seguridad.

Capítulo 8. Diseño de la salida

Este capítulo analiza los aspectos racionales que se encuentran detrás de los prototipos de sistemas, de los métodos de desarrollo de prototipos y como utilizarlos una vez que se desarrollan.

Título	Elementos y Herramientas en el Desarrollo de Sistemas de Información
Autor	Mario G. Piattini
Editorial	Addison Wesley Iberoamericana
Edición	España 1995
ISBN	0-287894-1
Colocación Biblioteca	QA76.758 E51
Comentario	
PARTE II. TÉCNICAS Y METODOLOGÍAS	
Capítulo 2. Técnicas y metodologías en el desarrollo de software	
En este capítulo se ahonda sobre los conceptos de términos como: <i>realidad original</i> ,	

modelo, sistema, método, proceso, metodología, etc. Menciona que un método es la secuencia de modelados que ayuda a construir, a partir de la realidad, una o varias cadenas de modelos, derivados de otros, con el objetivo de lograr un modelo material final o sistema que concrete la nueva realidad deseada en relación con el original. En sentido estricto, metodología es el discurso, ciencia o estudio de los métodos.

Incluye los elementos de los métodos, intentos de clasificación de métodos, encapsulación de los métodos y el uso de métodos y herramientas. También trata sobre el ciclo de desarrollo (amplia, pero impropriamente llamado ciclo de vida) tiene tres grandes grupos de fases:

- Preejecutor (planificación, especificación y análisis).
- Ejecutor (diseño, codificación, pruebas).
- Postejecutor (documentación, formación de usuarios, mantenimiento, rectificaciones).

Capítulo 4. Metodología MERISE

Este capítulo se centra en el nacimiento de la metodología llamada MERISE, menciona sus aportaciones, nace como metodología de análisis y diseño de sistemas de información, aportando un plan de trabajo y técnicas de modelado para la concepción de aplicaciones coherentes para el área de gestión de las empresas.

Se presentan las características más destacables de MERISE:

- Aproximación racional para la resolución de problemas.
- Visión cronológica y completa de las relaciones entre procesos.
- Coherencia de los sistemas resultantes.
- Amplio grado de difusión alcanzado por el método.

Se incluyen las etapas y fases de la metodología MERISE: esquema director, estudio previo, estudio detallado, estudio técnico, realización, mantenimiento.

Capítulo 6. Metodología de la ingeniería de la información

Este capítulo se enfoca en la ingeniería de la información que es ya una de las metodologías más populares, desarrollada inicialmente por James Martín, actualmente existen también otras metodologías inspiradas en ella.

Se incluyen las principales características de esta metodología:

- Proporcionar un entorno metodológico para cada fase e integrar estos con técnicas y herramientas.
- Uso de técnicas que están integradas en cada fase de la metodología.
- Ayudar a la organización a conseguir y defender ventajas competitivas.
- Proporcionar una arquitectura para el soporte integrado de la información

ejecutiva.

- Soportar las necesidades de información de gestión ejecutiva de una manera eficiente.
- Involucrar de una manera eficaz a los usuarios.
- Mejorar la calidad.
- Reducir tiempos y riesgos.

Las fases de la metodología son: planificación, análisis, diseño, construcción.

Capítulo 7. Metodología métrica versión 2

El contenido central de este capítulo es la metodología métrica versión 2 desde su historia y evolución. En métrica versión 2 se hace énfasis en la utilización de técnicas gráficas con el fin de describir los sistemas de forma sencilla y mejorar la comunicación entre los desarrolladores y los usuarios.

Las técnicas incluidas son:

- Diagramas de Flujos de Datos (DFD).
- Modelado de datos.
- Historia de la vida de entidades.
- Diseño estructurado.
- Entrevistas.
- Análisis coste beneficio.
- Pruebas.
- Factores críticos de éxito.
- Técnicas matriciales.

Capítulo 9. Técnicas y metodologías orientadas a objetos

El capítulo hace referencia a la orientación a objetos, sus técnicas y metodologías, a modo de introducción menciona todos los orígenes de la orientación a objetos, además presenta los conceptos básicos de la orientación al objeto, y a continuación, el origen de las metodologías orientadas al objeto y el proceso y técnicas del desarrollo orientado al objeto, así como, las características de las metodologías orientadas al objeto. También incluye las limitaciones de las metodologías actuales y tendencias futuras.

Título	Métodos Orientados a Objetos. Conceptos Fundamentales
Autor	James Martin, James I. Odell
Editorial	Prentice Hall
Edición	México 1997
ISBN	970-17-0045-7
Colocación Biblioteca	QA76.64 M368518

Comentario

Capítulo 1. Introducción

Existen muchos métodos para hacer desarrollos, siempre existirán y deberán existir. Las diversas tareas pueden tener características diferentes que requieren, desde luego, distintos tipos de métodos. El reto es la selección e integración de los procedimientos. El análisis orientado a objetos no es un enfoque que modele la realidad, sino la manera en que las personas la comprenden y procesan.

PRIMERA PARTE. LA BASE DE LA O. O., LA ESTRUCTURA DEL OBJETO

Capítulo 2. Conceptos

Este capítulo se centra en los conceptos, porque son importantes para nosotros, cómo los utilizamos y comunicamos y para qué sirven en el análisis orientado a objetos; los conceptos determinan y conforman nuestra realidad, a la vez que nos proporcionan estabilidad. Además presenta las nociones fundamentales, los objetos a partir de los cuales podemos construir diagramas conceptuales.

Capítulo 3. Objetos

El contenido de este capítulo se centra en los objetos, que son y como los percibimos, su clasificación, porque los conceptos son tan importantes para nuestra comprensión de los objetos, que significa que los objetos tengan un ciclo de vida. Un conjunto es una colección de los objetos a los que se aplica un concepto particular. Un objeto puede ser miembro de varios conjuntos en cualquier momento, lo que es; la clasificación múltiple. Además, la membresía de conjunto de un objeto puede cambiar con el tiempo; fenómeno denominado clasificación dinámica.

Título	Análisis Estructurado Moderno
Autor	Edward Yourdon
Editorial	Prentice Hall
Edición	México 1997
ISBN	968-880-3030-0

Comentario

PARTE I. INTRODUCCION

Capítulo 2. La naturaleza de los sistemas

Este capítulo trata acerca de los sistemas; su definición, diferencia entre sistemas naturales y sistemas hechos por el hombre, las cinco razones principales por las que

algunos sistemas no deben automatizarse. Los cinco componentes principales de un sistema de información automatizado típico: *hardware, software, personas, datos y procedimientos*.

Establece una división en categorías, más útil de los sistemas automatizados:

- Sistemas en línea
- Sistemas de tiempo real
- Sistemas de apoyo a decisiones
- Sistemas basados en el conocimiento

Capítulo 3. Los participantes en el juego de los sistemas

Este capítulo nos dice cuales son las categorías de personas con las que se interactúa a lo largo de un proyecto: usuarios, administración, auditores, analistas de sistemas, diseñadores de sistemas, programadores y personal de operaciones.

Enumera tres tipos de usuario:

- Usuario supervisor
- Usuario ejecutivo
- Usuario operacional

Estudia cuales serán sus reacciones, y la diferencia entre usuarios novatos y expertos, especifica cual es el papel de la administración y el analista en un proyecto de desarrollo de sistemas.

Capítulo 4. Herramientas del análisis estructurado

Este capítulo se centra alrededor de las herramientas del análisis estructurado. nos explica la naturaleza y componentes de:

- Diagrama de flujo de datos
- Diccionario de datos
- Especificación de proceso

Enseña como modelar: datos almacenados y relaciones entre datos, comportamiento dependiente del tiempo de un sistema y la estructura de un programa de computadora.

Capítulo 5. El ciclo de vida del proyecto

En este capítulo se explica de manera detallada lo que es el ciclo de vida de un proyecto, concepto, características, componentes y evalúa diferentes tipos de ciclo de vida.

Capítulo 7. Cambios en el análisis de sistemas

En este capítulo se exponen los problemas del análisis clásico de sistemas, la evolución que ha sufrido el análisis estructurado clásico, explica porque las herramientas automatizadas son importantes para el futuro del análisis de sistemas, y porque los problemas del análisis estructurado clásico han llevado a la elaboración de prototipos.

Título	Análisis y Diseño de Sistemas de Información
Autor	Jeffrey L. Whitten, Lonnie D. Bentley, Victor M. Barlow
Editorial	Mc Graw Hill
Edición	Tercera Edición España 1998
ISBN	84-8086-2524

Comentario

Capítulo 4. Técnicas y metodologías del desarrollo de sistemas

Este capítulo ofrece una introducción a las técnicas y metodologías del desarrollo de sistemas, un asunto independiente, aunque próximo, del ciclo de vida de desarrollo de sistemas.

En el capítulo primero, se distingue entre técnicas, metodologías y ciclo de vida, la mayor parte del capítulo se centra después en las denominadas "técnicas estructuradas". Ofrece una introducción a las cada vez más extendidas técnicas de prototipos y de diseño conjunto de aplicaciones.

Por último, presenta las metodologías disponibles comercialmente como una alternativa con la que cuentan las empresas que adquieren paquetes de técnicas y métodos de aplicación gradual.

Título	Object Oriented Methods (Métodos Orientados a Objetos)
Autor	Iam Graham
Editorial	Addison –Wesley Publishing Company
Edición	Segunda edición 1992
ISBN	0-201-56521-8

Comentario

Capítulo 1. Basic concepts (Conceptos básicos)

Este capítulo inicia mencionando de manera muy breve los antecedentes históricos de la programación orientada a objetos, así como el análisis y diseño orientado a objetos, aparecen también algunos conceptos fundamentales de lo que es la orientación a objetos como son: la abstracción, la encapsulación, el polimorfismo y la herencia.

Capítulo 7. Object-Oriented Design and Analysis (Análisis y Diseño Orientado a Objetos)

Este capítulo analiza los diferentes métodos de diseño orientados a objetos, de entre ellos los más comunes como son: Diseño Orientado a Objetos General (GOOD), Diseño Orientado a Objetos Jerárquico (HOOD), Diseño Estructurado Orientado a Objetos (OOSD) y Diseño Estructurado de Jackson (JSD) además de la metodología de Booch.

También explica cómo debe realizarse el análisis y después el diseño sobre la base de lo que proponen Coad y Yourdon.

Capítulo 8. Managing Object-Oriented Methods Administración de Métodos Orientados a Objetos)

Este capítulo se centra principalmente en la importancia que tiene el crear prototipos para administrar el análisis y diseño de una manera sencilla y controlada.

2.1.3 Libros de lectura rápida

Título	A Practical Guide to Enterprise Architecture
Autor	James McGovern, Scott W. Ambler, Michael E. Stevens, James Linn, Elias K. Jo, V. En ingles.
Editorial	Editado por Prentice Hall PTR
Comentario	
Este libro ayuda a los lectores a crear estrategias adaptativas para implantar arquitecturas organizacionales en forma exitosa. Presenta ejemplos desde la experiencia propia de los autores construyendo infraestructuras de software para empresas líderes en servicios financieros, telecomunicaciones, medios y comercio electrónico.	
Los autores proveen guía de principio a fin para construir arquitecturas efectivas de sistemas, de software y orientadas al servicio. Provee explicaciones comprensivas de conceptos y métodos de arquitectura organizacional, extiende el Rational Unified Process (RUP) para incluir arquitectura organizacional, unifica la visión de las distintas disciplinas arquitecturales para crear bosquejos de estrategia y presenta perspectivas sobre desarrollo organizacional y gestión de la tecnología.	

2.2 Tesis

Título	Documentación del Análisis y Diseño de Sistemas Orientados a Objetos con UML
Autor	Jenny Jiménez Herrada
Fecha	2001

Carrera profesional	Licenciatura en Informática
Universidad	UNAM, Facultad de Contaduría y Administración
Colocación Biblioteca	001 623 J1 2001
Comentario	
<p>El propósito general de este trabajo de tesis es plantear la importancia que tiene la documentación de los sistemas, en particular los orientados a objetos; y proponer el uso de una herramienta que facilite la documentación. Se resalta la necesidad de realizar una documentación clara y actualizada para el desarrollo de sistemas orientados a objetos.</p> <p>Se hace una breve reseña de la evolución de los paradigmas de desarrollo hasta llegar a la orientación a objetos y al UML, de la cual se mencionan sus principales creadores y múltiples aportadores.</p>	

Título	Diseño Orientado a Objetos Fundamentos y Aplicaciones
Autor	Olga Hernández Sánchez
Fecha	1995
Carrera profesional	Ingeniero en computación
Universidad	UNAM-Aragón
Colocación Biblioteca	Biblioteca Nacional 001-41-132-H4.1995
Comentario	
<p>El principal objeto que tiene esta tesis es dar a conocer todos los conceptos y aplicaciones en el diseño orientado a objetos basados en la metodología de Booch. Por lo cual este trabajo aporta varios elementos a nuestro tema de estudio.</p>	

Título	La Metodología de Orientación a Objetos como nueva herramienta para el Análisis y Diseño de Software
Autor	Adriana Santos Rosas
Fecha	1994
Carrera profesional	Ingeniero en computación
Universidad	UNAM – Facultad de Ingeniería
Colocación Biblioteca	Biblioteca Central 01-41132-S2-1994
Comentario	
<p>Este trabajo de tesis tiene como propósito principal el dar a conocer la metodología de</p>	

la orientación a objetos, como una herramienta que facilite el desarrollo de sistemas, se mencionan métodos de programación orientados a objetos, lenguajes orientados a objetos y bases de datos orientados a objetos.

También menciona las ventajas, desventajas y costos del paradigma orientado a objetos sobre todo cuando se desea realizar la migración del paradigma estructurado al orientado a objetos.

Título	Un Enfoque Formal a la Construcción de Software usando Eiffel
Autor	Nino Pineda Villalvazo
Fecha	1998
Carrera profesional	Licenciatura en informática
Universidad	UNAM – Facultad de Contaduría y Administración
Colocación Biblioteca	Biblioteca FCA 001-623-P1-1998
Comentario	
<p>En este trabajo de investigación se describe brevemente el campo de la ingeniería de software y una de sus áreas de investigación: los métodos formales, y a continuación se describe un método riguroso de construcción de software, dando así una posible transición entre los métodos tradicionales de desarrollo y los formales. Menciona los antecedentes de Eiffel, sus características, su contexto dentro de la orientación a objetos, y él porque se propone como un método riguroso para la construcción de software.</p>	
Título	Metodología para el Desarrollo de Software: Diccionario de Términos Administrativos
Autor	Marisela Barrios Vázquez
Fecha	1997
Carrera profesional	Licenciatura en informática
Universidad	UNAM – Facultad de Contaduría y Administración
Colocación Biblioteca	Biblioteca FCA 001-623-B1-1997
Comentario	
<p>En esta tesis se propone la creación de un diccionario electrónico de términos administrativos que permita la búsqueda de información de manera eficiente, aunado al contexto de la <i>metodología de desarrollo de software</i> con el principal objetivo de</p>	

demostrar que por muy pequeño que sea un sistema es necesario llevar a cabo un análisis para su desarrollo.

Título	UML y RUP como elementos de Sistemas Orientados a Objetos
Autor	Silvia Jiménez Luna, Claudia Elvira Soriano
Fecha	2002
Carrera profesional	Licenciatura en informática
Universidad	UNAM – Facultad de Contaduría y Administración
Colocación Biblioteca	Biblioteca FCA 001-623-J1-2002
Comentario	
<p>A través de esta tesis se pretende dar a conocer los puntos clave que se consideran como pilares fundamentales para lograr el desarrollo de sistemas orientados a objetos exitosos:</p> <ul style="list-style-type: none"> • <i>Un proceso de desarrollo que defina la relación entre las tareas.</i> • Un lenguaje unificado que permita un análisis y diseño uniforme que mejore la comunicación entre usuarios y desarrolladores. • Personas con formación y capacitación necesaria. 	

2.3 Conferencias

Título	La Crisis del Software
Organizador	Dirección General de Servicios de Cómputo Académico UNAM.
Fecha de realización	28 de Noviembre
Expositor	Jesús Zavala
Material	Diapositivas
Comentario	
<p>La conferencia tuvo como objetivo principal hacer hincapié en la importancia que tiene actualmente la ingeniería de software al aplicar técnicas, métodos y herramientas correctas, ya que es necesaria para desarrollar software confiable, extensible, seguro y complejo. Dio a conocer las causas estructurales de <i>la crisis del software</i> y su solución.</p>	

Principales causas del software malo:

- El desarrollo de software se ve más como una actividad artesanal que como una disciplina de ingeniería.
- El enfoque en la educación y capacitación ha propiciado la “mentalidad de artesano”, ni siquiera arte.
- Los errores y deficiencias en el software se intentan corregir con “toneladas de capacitación”.

Esto implica que haya :

- Productos pobres: modelado, código.
- Procesos pobres: metodologías, procedimientos.
- Personas pobres: “mentalidad artesanal”, deficiente.

¿La crisis del software tiene solución?

- Aplicar un cambio drástico en la forma de desarrollar software (Reingeniería en el desarrollo).
- Aplicar metodologías, técnicas y métodos que garanticen predicibilidad (metodologías, ciencia, Project Management).
- En resumen, ingeniería de sistemas, ingeniería de software y conocimiento organizacional.

En la conferencia el expositor mencionó las siguientes cuestiones que son practicadas actualmente en el desarrollo del software:

- El diseño, documentación y evaluación “son de importancia secundaria” y se completan u omiten después de terminar porque “es desperdiciar el tiempo, lo importante es programar”.
- Las estadísticas muestran que el enfoque ad hoc no funciona y sin embargo se utiliza ampliamente .
- Este hecho respalda nuestra hipótesis, ya que nosotras establecemos que el uso de una adecuada metodología es fundamental para el eficiente desarrollo de software.

2.4 Diplomados

Título	Diplomado en Ingeniería de Software
Lugar	Fundación Arturo Rosenblueth
Duración	15 hrs.
Sede	Ciudad de México

Comentario

Este curso satisface los requerimientos teórico-prácticos para el desarrollo de software; comenzando con los fundamentos de esta disciplina, así mismo, cubriendo los aspectos de modelado de procesos de negocio, el paradigma de orientación a objetos (UML y el Proceso Unificado de Desarrollo), la administración de proyectos de software, la reutilización del mismo y su administración de calidad.

Título	Diplomado en Sistemas de Información
Lugar	Fundación Arturo Rosenblueth
Duración	15 hrs.
Sede	Ciudad de México

Comentario

En este curso se obtendrán conocimientos tanto teóricos como prácticos sobre ingeniería de procesos de negocios, análisis y diseño de sistemas de información, administración de la información y del conocimiento, sistemas de información distribuidos, integración de soluciones tecnológicas en las organizaciones, cambio tecnológico en las organizaciones.

3 MARCO CONCEPTUAL

3.1 Antecedentes de las metodologías de desarrollo de software

El antecedente de las metodologías de desarrollo de software se remonta a los años de la década de 1950, cuando el desarrollo era “artesanal”¹, no existían las metodologías de desarrollo de software y los desarrolladores eran programadores. Se tenían como principales problemas que los resultados eran impredecibles, no se controlaba el proyecto, los cambios organizativos afectaban al proceso de desarrollo de software.

En 1960, la programación era substancialmente “*ad hoc*”², se trato de un gran avance conseguir que un programa fuese procesado. Se construían, sin embargo, algunos sistemas complejos, pero su construcción era totalmente empírica o se trataba de una actividad virtuosa. El énfasis recaía, sobre pequeños programas, que era todo lo que se podía manejar de una forma predecible. Fue entonces cuando se empezó a comprender que los sistemas informáticos no eran solamente procesadores de números, sino también procesadores de información en forma simbólica.

3.1.1 Origen de las metodologías de desarrollo de software

El origen de las metodologías de desarrollo de software es el desarrollo estructurado, el cual es una transición del desarrollo artesanal a metodológico. Podemos decir que para finales de la década de 1960 e inicios de la década de 1970 el análisis estructurado surge de la necesidad de buscar una forma interpretativa más rápida y eficiente, de tal manera que se pudiesen definir los requerimientos del usuario y las especificaciones funcionales del sistema ya que para esto existían documentos textuales.

El análisis estructurado fue la metodología de análisis de sistemas que tuvo uno de los impactos más grandes en el ambiente de producción del software. Desde sus orígenes a mediados de la década de 1970, con los libros de Tom Demarco [Demarco 1979]³ y, Chris Gane y Trish Sarson [Gane 1979]⁴, luego durante la década siguiente se extendió rápidamente hasta volverse en una *metodología estándar*. En los años de la década de 1980, muchos autores trabajaron en su modernización y en la incorporación de nuevas herramientas y estrategias de modelamiento.

¹ Aún cuando la palabra artesano significa adj. y s. Persona que ejercita un arte manual, en la jerga informática el termino artesanal se refiere a la forma como en un principio se desarrollo el software mediante ciertas técnicas de producción y gestión para una fabricación rutinaria.

² Expresión latina. Ad hoc: Significa para un fin determinado, especialmente.

³ Tom DeMarco, "Structured Analysis and Systems Specification", Prentice-Hall, Inc., Englewood Cliffs, N.J., 1979.

⁴ Chris Gane and Trish Sarson, "Structured Systems Analysis: Tools and Techniques", Prentice-Hall, Inc., Englewood Cliffs, N.J., 1979.

El origen de las metodologías de desarrollo de software ocurre dentro del ámbito de la informática, cuando su historia nos presenta situaciones, en las que una etapa es superada merced a invenciones revolucionarias.

Las metodologías de desarrollo de software vienen a superar, más específicamente, la forma en cómo se trabajaba para la construcción de software. Cada una de las metodologías de desarrollo de software que se originan, vienen a renovar los métodos ya agotados con que se desarrollaba el software, además de cubrir nuevas necesidades.

El uso de las metodologías de desarrollo de software, llega, en el momento en que confluyen dos factores que bien vale la pena destacar:

- La complejidad que los sistemas informáticos pueden tener, debido al progreso que ha tenido el hardware gracias a la potencialidad de la tecnología y sus creadores.
- La necesidad de que estos complejos sistemas de software funcionen correctamente y de manera eficiente.

Complejidad de los sistemas de información:

Como ya se ha mencionado anteriormente la complejidad en los sistemas de información es un factor que ha desencadenado el uso de las metodologías de desarrollo de software, en la actualidad existen varios tipos de complejidad:

- Complejidad arbitraria. Es a la que se enfrenta un programador solitario, por ejemplo simular una calculadora.
- Complejidad industrial. Cuando están involucrados más de un programador, implica muchos algoritmos. Una característica distintiva del software industrial es que resulta sumamente difícil para el desarrollador individual comprender todas las sutilezas de su diseño. Lamentablemente, parece ser una propiedad esencial de todos los sistemas de software de gran tamaño.
- La complejidad del dominio del problema. Surge habitualmente entre los usuarios del sistema y sus desarrolladores; los usuarios suelen encontrar grandes dificultades al intentar expresar sus necesidades de tal manera que los desarrolladores las entiendan.
- Complejidad surgida durante el desarrollo del software. Esta complejidad se refiere a que los requisitos de un software cambian frecuentemente durante su desarrollo, especialmente porque la mera existencia del proyecto altera las reglas del problema.
- Complejidad de evolución con respecto al tiempo. Por ejemplo el problema del año 2000 (Y2K).

3.2 Visión histórica de las metodologías de desarrollo de software

El origen que tiene las metodologías de desarrollo de software va muy ligado también a la evolución que se ha tenido del hardware, ya que sin éste, simplemente no habría forma de que se pudiera construir software, y no existirían las metodologías para poder desarrollarlo. Por ello se presenta una breve evolución cronológica del tratamiento mecánico, electromecánico y electrónico de la información, aunado al origen y evolución de las metodologías de desarrollo de software.

Año	Suceso / Avance	Reseña (Historia de las computadoras)
5000 a.C.	"Ábaco". Mesopotamia	El ábaco es considerado como el más antiguo aparato de cálculo. Del griego abax (polvo). Tiene su origen en Mesopotamia después pasa a Grecia, que a su vez, lo da a conocer a los romanos, y a algunos países de Asia tales como: China, Rusia y Japón. El ábaco esta formado por un marco rectangular y piezas móviles llamadas cuentas que se deslizan sobre los ejes.
Distintos tipo de ábaco <ul style="list-style-type: none"> • Ábaco griego o Abax • Ábaco ruso o Schiotti • Ábaco chino o Suan Pan • Ábaco japonés o Soroban 		En diversas ocasiones se han organizado concursos de cálculo en los que competían el ábaco y calculadoras mecánicas y electromecánicas, y el ábaco ha logrado imponerse a sus contrincantes en muchas operaciones, incluidos los cálculos complejos. Actualmente es utilizado en algunos lugares de Asia.
1550-1617	"Huesos de Napier". John Napier, matemático escocés.	Se le considera el principal inventor de los logaritmos y también descubre varias fórmulas de trigonometría esférica y la resolución de triángulos esféricos. En 1614 publica su magna obra; las primeras tablas de logaritmos, en el libro "Rabdología". Los "Huesos o Rodillos de Napier" pueden hacer operaciones de multiplicar o dividir. El producto de dos números cualesquiera aparecía sobre los rodillos de madera, por lo que los números tenían que ser escritos o memorizarlos.
Dispositivos mecánicos de cálculo		
Simbólicamente, anticipan el cambio tecnológico. Son un símbolo porque no alcanzan una aplicación generalizada o productiva. Se trata de una conquista que inicia el proceso de automatización del tratamiento de la información. Y se consolida con la creación de algunos prototipos realmente ingeniosos, cuya excelente concepción formal se ve limitada por las dificultades técnicas de su realización.		
Hacia 1592-1623	"Primera calculadora". Wilhelm Schickard, científico alemán.	Wilhelm Schickard es quién diseña y construye lo que podemos considerar como la primera máquina mecánica de calcular basada en ruedas dentadas, ya que podía efectuar las cuatro operaciones aritméticas básicas: suma, resta, multiplicación y división. Shickard construía esta máquina para el matemático alemán Johannes Kepler (1571-1630). Desafortunadamente esta primera pieza mecánica de cálculo fue destruida en un incendio y no pudo ser reconstruida, ya que Shickard muere debido a la peste, durante la <i>Guerra de los Treinta Años</i> . A pesar de que pertenece a este hombre la paternidad de la primera calculadora, la historia ha registrado a Pascal como su inventor.
1623-	"Pascalina"	Blaise Pascal (1642). Filósofo matemático y físico francés,

1662	Blaise Pascal	<p>construyó su máquina cuando sólo tenía diecinueve años, y lo hizo con la intención de ayudar a su padre, que era agente de impuestos, y para liberar a los hombres de la servidumbre de realizar cálculos repetitivos.</p> <p>Primera calculadora mecánica conocida hasta fechas recientes, era una máquina basada en ruedas dentadas interrelacionadas; la rotación completa de ellas hacía girar un paso a la rueda siguiente. La pascalina sólo realizaba sumas y restas.</p>
<p>Las calculadoras de Pascal, Morland y Leibnitz chocan con un gran inconveniente técnico. El desarrollo tecnológico de los siglos XVII y XVIII no estaba a la altura de sus diseños. La producción artesanal no alcanzaba el grado de precisión que se requería para conseguir un ajustado funcionamiento de la maquinaria.</p> <p>La revolución industrial se haría esperar más de un siglo y, con ella, la posibilidad de la producción en serie de maquinaria precisa. No obstante, ahí estaban los prototipos de lo que habría de convertirse en una larga tradición de calculadoras mecánicas.</p>		
1625-1695	Aritmómetro Samuel Morland	Es la primera máquina de calcular que se conoce, parece haber sido la modificación de la construida por Pascal. Consiste en una serie de ocho ruedas que giran alrededor de sus ejes.
1646-1716	"Calculadora universal" Gottfried Leibnitz, matemático alemán.	Partiendo de los modelos de Pascal y Samuel Morland, Leibnitz ideó un ingenioso mecanismo para conseguir que su calculadora realizara, además de sumas y restas, multiplicaciones y divisiones e incluso fuera capaz de extraer raíces cuadradas. Este mecanismo fue bautizado con el nombre de "Rueda escalada de Leibnitz".
<p style="text-align: center;">El origen de la programación</p> <p>Curiosamente, su origen no se halla en el cálculo, sino en la industria textil. Se trata del telar automático diseñado por el francés Joseph Marie Jacquard (1752-1834). El telar de Jacquard tiene un efecto indirecto pero eficaz en el desarrollo de la prehistoria de las computadoras modernas. La máquina tejía de forma automática, siguiendo un patrón o "programa" de trabajo. De esta manera producía tejidos según el modelo que se le había proporcionado y se ajustaba a él escrupulosamente. El secreto estaba en unas tarjetas perforadas. Estas determinaban el dibujo en el tejido ya que se introducían a modo de instrucciones.</p> <p>Esta innovación técnica realizaba tres aportaciones teóricas muy importantes:</p> <ul style="list-style-type: none"> • Un modelo de automatización. • La codificación de la información. • La programación de las instrucciones. <p>Que siguen vigentes, tanto en el medio original para el que fue desarrollado como en los "cerebros electrónicos".</p>		
1791-1871	"Máquina de diferencias". Charles Babbage.	<p>En Inglaterra, Charles Babbage, profesor de matemáticas de la <i>Universidad de Cambridge</i>, trabajaba hacia 1822 en un proyecto financiado por la <i>Royal Society</i> al cual llamó la "máquina diferencial", capaz de calcular polinomios de sexto grado y tabular mecánicamente hasta veinte cifras y ocho decimales. Gracias a ella, el propio inventor confecciona tablas numéricas y publicó las tablas de logaritmos de números naturales.</p> <p>La máquina nunca fue terminada debido a que, mientras avanzaba en la construcción constantemente se le ocurrían mejoras para perfeccionar el aparato. En 1833 abandona el primer proyecto, e inspirado en las tarjetas perforadas de Jacquard se propone realizar el verdadero sueño de su vida: la "máquina analítica".</p>
1833	"Máquina analítica" Charles Babbage.	Es la precursora de la computadora del siglo XX. Consiste en una máquina de calcular de manera polivalente, con la capacidad de operar de formas distintas en virtud al tipo de problema que se

		<p>planteará. Esta definición recurre al concepto de una máquina de propósito general.</p> <p>Esta calculadora, que sería capaz de realizar cualquier tipo de cálculo de manera digital, tampoco pudo ser construida debido a que su tecnología era muy adelantada para su época y nunca pudo construir las sofisticadas piezas que diseñaba para ella.</p>
<p>En la máquina analítica hallamos los elementos básicos de la computadora actual:</p> <ul style="list-style-type: none"> • Mecanismos de entrada: Tarjetas perforadas tanto para datos como para instrucciones. • Memoria: Consistía en un conjunto de mil columnas con cincuenta ruedas cada una para "almacenar" mil números de cincuenta cifras. • Unidad de control: Mecanismo que rige la realización de las operaciones según un orden necesario, tal como esta escrito en el programa. • Unidad aritmético-lógica: Realiza las operaciones de cálculo numérico y discriminaciones lógicas. • Mecanismos de salida: Donde se leían los resultados de los cálculos. 		
1815-1853	Augusta Ada, matemática.	<p>Condesa de Lovelace e hija del poeta Lord Byron y aficionada amiga de Babbage, se interesó mucho en la máquina analítica y trabajo junto con él. Está considerada como la primera programadora pues escribió secuencias de instrucciones en tarjetas perforadas, inventó métodos de <i>programación</i> como la <i>subrutina</i> e introdujo en sus programas las <i>iteraciones</i> y el <i>salto condicional</i>, lo que abre ya la posibilidad de tomar decisiones automáticamente. Otra de sus notables contribuciones es que propuso utilizar el sistema binario en lugar del decimal que utilizaba Babbage, para la codificación de los programas en tarjetas perforadas.</p>
Dispositivos electromecánicos de cálculo		
1860-1929	Dr. Herman Hollerith	<p>En 1886, el Dr. Herman Hollerith, estadístico empleado en la oficina de censos de USA, desarrolló un sistema basado en tarjetas perforadas para codificar los datos de la población en el censo de 1890, ya que el resultado del censo de 1880 se había entregado nueve años después de iniciado. Con el método de Hollerith el nuevo censo se terminó en poco menos de tres años.</p> <p>Esta máquina censal ya es eléctrica y contiene componentes electromecánicos, aunque está dedicada sólo a procesos de censos.</p>
1873-1961	"Tubo de vacío (bulbo)". Lee De Forest.	<p>Lee De Forest inventa el tubo de vacío (bulbo) de tres elementos, que más tarde tiene una gran importancia en el desarrollo de las computadoras.</p>
1919	"Flip-flop o basculador". W. H. Eccles y F. W. Jordan.	<p>En 1919 W. H. Eccles y F. W. Jordan descubren el <i>flip-flop</i> o basculador, un circuito binario capaz de asumir uno de dos estados estables.</p>
1916	"Información". Claude E. Shannon, matemático estadounidense.	<p>Claude E. Shannon (nacido en 1916), creador de la moderna teoría de la información, la define como: "Información es todo lo que reduce la incertidumbre entre diversas alternativas posibles". Shannon fue el creador del término bit, y demostró que el <i>Álgebra Boole</i> (George S. Boole (1815-1864) publica en 1854 su tratado de álgebra en donde utiliza los operadores lógicos AND, OR y NOR) es la herramienta más adecuada para estudiar los sistemas binarios y, por supuesto, su aplicación en la operación de las computadoras.</p>

Generaciones de computadoras		
<p>El avance de la tecnología de las computadoras, a partir de los primeros años del siglo XX ha sido sorprendente. El descubrimiento de los nuevos dispositivos electrónicos, los grandes avances de la programación y el acelerado desarrollo de los nuevos <i>sistemas operativos</i>, han marcado fechas que permiten clasificar a las computadoras de acuerdo a sus componentes y a su capacidad de procesamiento, agrupándolas por <i>generaciones</i>:</p>		
Primera generación		
<p>No se considera una fecha de inicio, ya que los trabajos para construir estas primeras computadoras comenzaron con la máquina analítica de Babbage. En cambio sí se puede determinar aproximadamente la fecha final de esta etapa en la década de los cincuenta, ya que en 1947 se descubre el primer transistor (<i>Transfer Resistance</i>), elemento que dio origen a las primeras computadoras de la segunda generación.</p>		
1944	"MARK I". Howard Aiken	Se caracterizaron por estar constituidas de relevadores "relés", se consigue llevar a buen término el proyecto de la primera computadora electromecánica, iniciado en 1939 y cuyo nombre en el proyecto oficial era ASCC (Automatic Sequence Controller Calculator). Con grandes dimensiones y bastantes elementos.
1946	"ENIAC". Mauchly y Eckert	Se presenta el proyecto del ENIAC (Electronic Numerical Integrator and Computer), su primera utilización fue para la construcción de tablas para el cálculo de trayectoria de proyectiles.
1952	"MANIAC I". "MANIAC II".	Se construyen los ordenadores MANIAC-I y MANIAC-II, con lo que se termina la prehistoria de la informática.
Desarrollo convencional		
<p>No existen metodologías de desarrollo. Desarrollo artesanal (década de 1950).</p> <ul style="list-style-type: none"> • Resultados impredecibles. • No hay control del proyecto. • Los cambios afectan severamente al proyecto. 		
Segunda generación		
<p>La característica principal en cuanto a los equipos (<i>hardware</i>) es la inclusión de transistores. Respecto a la programación o software, sigue dominando los sistemas de tarjeta o cinta perforada para la entrada de datos.</p>		
1954	"TRADIC".	John Bardeen, Walter H. Brattain y William Shockley diseñan el primer transistor, logran avances muy significativos como la construcción en 1954, de la primera computadora transistorizada, la TRADIC (TRAnsistorized Airborne DIgital Computer), que ya incluía 800 transistores en su estructura interna.
1954-1957	"FORTRAN".	Otro gran logro de esta época es el desarrollo del primer lenguaje de alto nivel, el FORTRAN (FORmula TRANslator). John Backus y algunos de sus colaboradores, empleados de IBM, empezaron este proyecto en 1954 y lo presentan formalmente en 1957. El lenguaje FORTRAN es muy apropiado para trabajos científicos, matemáticos y de ingeniería.
1958	"LISP".	John McCarthy desarrolla el lenguaje LISP (acrónimo de LIST Processor), que aporta grandes avances en la investigación sobre <i>Inteligencia Artificial</i> por la facilidad con que permite el manejo de símbolos y listas.
1959-1960	"COBOL".	COBOL (COmmon Business Oriented Language), que representa uno de los más grandes avances en cuanto a <i>portabilidad</i> de los programas entre diferentes computadoras; es decir, es uno de los primeros programas que se pueden ejecutar en diversos equipos de cómputo después de un sencillo procesamiento de compilación.

	"Computadoras construidas con transistores".	con	IBM 1401 Las Honeywell 800 y su serie 5000 UNIVAC M460 Las IBM 7090 y 7094 NCR 315 Las RCA 501 y 601 Control Data Corporation con su conocido modelo CDC1604 <i>Las supercomputadoras UNIVAC LARC, e IBM Stretch (1961).</i>
1955-1965	"Programación de cualquier modo".	de	Programas pequeños, representación de información estructurada y simbólica, comprensión elemental de diagramas de flujo, las especificaciones van en forma de prosa, no hay gestión, ensambladores.
Tercera generación			
El siguiente paso fue la integración a gran escala de transistores en <i>microcircuitos</i> llamados <i>procesadores o circuitos integrados monolíticos</i> LSI (<i>Large Scale Integration</i>), así como la proliferación de lenguajes de alto nivel y la introducción de programas para facilitar el control y la comunicación entre el usuario y la computadora, denominados <i>sistemas operativos</i> . El descubrimiento en 1958 del primer <i>Circuito Integrado (Chip)</i> , por el ingeniero Jack S. Kilby (nacido en 1928) de Texas Instruments, así como los trabajos que realizaba por su parte, el DR. Robert Noyce de Fairchild Semiconductors, acerca de los circuitos integrados, dieron origen a la tercera generación.			
1964	"IBM 360".		IBM marca el inicio de esta generación, cuando el 7 de abril de 1964 presenta la impresionante IBM 360, con su tecnología SLT (<i>Solid Logic Technology</i>).
1964	"Supercomputadora CDC 6600".		Control Data Corporation presenta la supercomputadora CDC 6600, que se consideró como la más poderosa de las computadoras de la época, ya que tenía la capacidad de ejecutar unos 3 000 000 de instrucciones por segundo (<i>mips.</i>).
Se empiezan a utilizar los medios magnéticos de almacenamiento, como cintas magnéticas de 9 canales, enormes discos rígidos, etc. Algunos sistemas todavía usan las tarjetas perforadas para la entrada de datos, pero las lectoras de tarjetas ya alcanzan velocidades respetables.			
1965-1975	"Programación de pequeña escala".	a	Algoritmos y programación, estructuras y tipos de datos, programas que se ejecutan una vez y terminan, especificaciones simples de entrada y salida, pequeño espacio simple de estados, lenguajes de programación, compiladores.
1967	"Algoritmos de estructuras de datos". Knuth	y	Se muestra la utilidad de pensar en ellos en forma independiente de los programas que los podían implementar.
1968	"Ingeniería de software".	del	"Ingeniería del Software es el estudio de los principios y metodologías para desarrollo y mantenimiento de sistemas de software" ⁵ .

⁵ Zelkowitz, M. V., Shaw, A. C. y Gannon, J. D.: *Principles of Software Engineering and Design*. Prentice-Hall, Englewoods Clif, 1979.

Desarrollo estructurado: De artesanal a metodológico.		
Programación estructurada. <ul style="list-style-type: none"> • Se define forma estática y dinámica de un programa. • Técnicas de programación estructurada. • Diseño estructurado • Primeros conceptos sobre diseño estructurado. • Nivel de abstracción más amplio. • Se refina el concepto de modularidad. • Análisis estructurado • Al principio se hace la descripción narrativa de los requisitos. • Principales problemas: monolíticas, redundantes, ambiguas, imposibles de mantener. • Se incorporan gráficos, se permiten particiones y son mínimamente redundantes. 		
1968	"Programación estructurada".	La programación estructurada es una técnica orientada a procesos para el diseño y la escritura de programas con mayor claridad y consistencia. Algunos de los primeros defensores de los principios de la programación estructurada fueron Corrado Böhm, Giuseppe Jacopini, J. Edgar Dykstra y Harland Mills.
Cuarta generación		
Esta marcada por la aparición del primer <i>microprocesador</i> en 1971, <i>Intel Corporation</i> , presenta el primer <i>microprocesador o Chip</i> de 4 bits, que en un espacio de aproximadamente 4 X 5 mm. contenía 2 250 transistores.		
1977-1984	"Microcomputadoras".	En 1977 aparecen las primeras microcomputadoras entre las cuales, las más famosas fueron las fabricadas por: <ul style="list-style-type: none"> • Apple Computer • Radio Shack • Commodore Business Machines • IBM se integra al mercado de las microcomputadoras con su Personal Computer de donde les ha quedado como sinónimo el nombre de PC, y lo más importante; se incluye un sistema operativo estandarizado, el MS-DOS (Microsoft Disk Operating System).
Quinta generación		
A mediados de la década de 1980 se estableció la quinta generación con base en los grandes acontecimientos tecnológicos en materia de microelectrónica y computación (software) como inteligencia artificial, sistemas expertos, redes neuronales, teoría del caos, algoritmos genéticos, fibras ópticas, telecomunicaciones, etc.		
1982	"Supercomputadora Cray". Seymour Cray.	Creación en 1982 de la primera supercomputadora con capacidad de proceso paralelo, diseñada por Seymour Cray, quien ya experimentaba desde 1968 con supercomputadoras.
Las computadoras de esta generación contienen una gran cantidad de microprocesadores trabajando en paralelo y pueden reconocer voz e imágenes. También tienen la capacidad de comunicarse con un <i>lenguaje natural</i> e irán adquiriendo habilidad para tomar decisiones con base en procesos de aprendizaje fundamentados en <i>sistemas expertos e inteligencia artificial</i> .		
El almacenamiento de información se realiza en dispositivos magnéticos ópticos con capacidades de decenas de gigabytes; se establece el DVD (Digital Video Disk o Digital Versatile Disk) como estándar para el almacenamiento de video y sonido; la capacidad de almacenamiento de datos crece de manera exponencial.		

Sexta generación

La sexta generación de computadoras está en marcha desde principios de los años de 1990, las computadoras de esta generación cuentan con arquitecturas combinadas paralelo / vectorial, con cientos de microprocesadores vectoriales trabajando al mismo tiempo; se han creado computadoras capaces de realizar más de un millón de millones de operaciones aritméticas de punto flotante por segundo (teraflops); las redes de área mundial (Wide Area Network, WAN) seguirán creciendo desorbitadamente utilizando medios de comunicación a través de fibras ópticas y satélites, con anchos de banda impresionantes. Las tecnologías de esta generación ya han sido desarrolladas o están en ese proceso. Algunas de ellas son: inteligencia artificial distribuida; teoría del caos, sistemas difusos, holografía, transistores ópticos, etc.

Diseño estructurado

Es una técnica orientada a procesos utilizada para fragmentar un programa grande en un conjunto jerarquizado de módulos y obtener un programa informático más fácil de implantar y mantener. Los principales defensores del diseño estructurado son Larry Constantine, Ed Yourdon, Jean Dominique Warnier, Ken Orr y Michael Jackson.

1975	"Diseño estructurado".	Primeros conceptos sobre diseño estructurado de Myers, Yourdon y Constantine, Page-Jones.
1975-1985	"Programación a gran escala".	Interfaces, estructuras de gestión, bases de datos de larga duración, programas ejecutándose continuamente, sistemas con especificaciones complejas, herramientas de entorno, documentos.

Análisis estructurado

Divide un sistema en procesos, entradas, salidas y archivos. Elabora modelos del tipo entrada-proceso-salida orientados a flujos para un problema o una solución de empresa. Los principales impulsores del análisis estructurado son Tom DeMarco, Chris Gane, Trish Sarson y Ed Yourdon. El análisis estructurado surge de la necesidad de buscar una forma interpretativa más rápida y eficiente, de tal manera que se pudiesen definir los requerimientos del usuario y las especificaciones funcionales del sistema. Pero esto no se daba porque lo que existía eran grandes volúmenes de información que había que leer por completo y que acarrearaban una serie de problemas de: monolitismo, redundancia, ambigüedad e imposibilidad de mantener.

El análisis estructurado surge de la necesidad de buscar una forma interpretativa más rápida y eficiente, de tal manera que se pudiesen definir los requerimientos del usuario y las especificaciones funcionales del sistema. Pero esto no se daba porque lo que existía eran grandes volúmenes de información que había que leer por completo y que acarrearaban una serie de problemas de: monolitismo, redundancia, ambigüedad e imposibilidad de mantener.

1977	"Análisis estructurado".	Primeros autores en el análisis Gane y Sarson.
1978	"Análisis estructurado MERISE".	Aportaciones de Weinberg y DeMarco al análisis estructurado. Nace la metodología MERISE.
1981	"SSADM".	Primera versión de Structured System Analysis and Design Method (SSADM).
1989	"MÉTRICA".	Primera versión de la metodología MÉTRICA.
1989	"Análisis estructurado Moderno". Ed Yourdon	Introduce una versión mejorada del análisis estructurado. Este método elimina la modelización detallada del sistema actual. Se introducen mejoras sobre todo para modelar sistemas de tiempo real y relaciones de situaciones complejas. Aumentando por ende la comunicación entre el analista y el sistema.
1985	"Sistemas de Tiempo Real". Ward & Mellor	Análisis y diseño estructurado para sistemas de tiempo real. Proponen incluir toda la especificación de control, en los Diagramas de Flujo de Datos (DFD's).
1987	"Sistemas de Tiempo Real". Matley & Philbhay	Análisis y diseño estructurado para sistemas de Tiempo real. Su método se basa en eliminar de los DFD's todo lo relativo a información de control.
1989	"Ingeniería de	Es una técnica basada en los datos, pero también sensible a los

	Información".	procesos que se aplica a las organizaciones consideradas en su conjunto más que a proyectos circunstanciales concretos.
Desarrollo orientado a objetos		
Se tratan procesos y datos de forma conjunta Nacen los lenguajes orientados a objetos Las técnicas estructuradas han servido de base para muchas metodologías orientadas a objetos		
1967	"SIMULA 67". Nygard y Pahl	Este lenguaje fue el primero en introducir los conceptos de clases, co-rutinas, y subclase, como los lenguajes orientados a objetos actuales.
1980	"Smaltalk 80".	Primer lenguaje, completo y robusto, orientado a objetos.
1986	"Diseño Orientado a Objetos (OOD)". Booch	El método no es secuencial, el desarrollo del diseño es evolutivo, incremental e iterativo.
1991	"Técnica de Modelado de Objetos (OMT)". Rumbaugh	Consiste en tres puntos de vista diferentes pero relacionados entre sí: El modelo de objetos, el modelo dinámico y el modelo funcional.
1992	"Ingeniería del Software Orientado a Objetos (OOSE)". Jacobson	Tiene un enfoque llamado orientado a casos de uso. En este enfoque un modelo de casos de uso sirve como un modelo central del que todos los modelos se derivan.
Diseño de prototipos		
Es una popular técnica de ingeniería utilizada para desarrollar modelos a escala de un producto o de sus componentes. A mediados de la década de 1980 aparecen las herramientas de generación de prototipos que son considerados como una alternativa al análisis estructurado para los usuarios.		
Desarrollo rápido de aplicaciones		
Es una combinación de diversas técnicas estructuradas (en especial la ingeniería de información basada en datos) con técnicas de prototipos y de desarrollo conjunto de aplicaciones cuyo fin es acelerar el desarrollo de sistemas.		

3.3 Fundamentos de las metodologías de desarrollo de software

El American National Standards Institute (ANSI) define a un *sistema* como: “Un conjunto de métodos, procedimientos o técnicas unidas para una interacción regulada para formar un todo organizado”. En general, podemos decir que un *sistema* es un conjunto de elementos que interactúan entre sí y que realizan una función específica para lograr un objetivo común.

Definición de “Sistema de Información (SI)- Es el medio por el cual los datos fluyen de una persona o departamento hacia otros y puede ser cualquier cosa, desde la comunicación interna entre los diferentes componentes de la organización y líneas telefónicas hasta sistemas de cómputo que generan reportes periódicos para varios usuarios. Los sistemas de información proporcionan servicio a todos los demás sistemas de una organización y enlazan todos sus componentes de tal forma que éstos trabajen con eficiencia para alcanzar el mismo objetivo”⁶.

“Sistema de información: Conjunto de elementos que están organizados para procesar información y cumplir un objetivo predefinido”⁷.

Elementos de un SI:

Personas, datos e información, Recursos (computacionales: software, hardware, redes).

Finalidad de los SI: Procesar entradas, mantener archivos de datos relacionados con la organización y producir información, reportes y otras salidas, para aumentar el conocimiento del usuario, o reducir su incertidumbre.

“Tipos de SI”⁸:

- Sistemas para el procesamiento de transacciones (TPS).
- Sistemas de información administrativa (MIS).
- Sistema para el soporte de decisiones (DSS).
- Sistemas Expertos (ES).

⁶ Seen James A., Análisis y Diseño de Sistemas de Información. Mc. Graw Hill, México 1998.

⁷ Si bien un sistema de información no es necesariamente automatizado, actualmente la idea es aplicar herramientas de cómputo que lo hagan más eficiente. De hecho muchos utilizan el término de Sistema Informático.

⁸ Los tipos de sistemas de información no totalmente excluyentes, pero es importante identificarlos para darles el tratamiento correspondiente.

Sistema de Información	Definición	Actividades
Sistema para el procesamiento de transacciones (TPS).	Su finalidad es mejorar las actividades rutinarias de una empresa. Procesan datos referentes a las transacciones. Una transacción es cualquier suceso o actividad que afecta a toda la organización.	Procesan datos con la finalidad de: registro, clasificación, orden, cálculo, sintonización o despliegue de los resultados.
Sistema de información administrativa (MIS).	Dan soporte en situaciones de <i>decisiones bien estructuradas</i> y mediante ellos es posible anticipar los requerimientos de información más comunes para que los administradores conozcan de antemano los factores que deben tener en cuenta para la toma de decisiones así como las variables con influencia más significativa sobre el resultado de una decisión.	Proporcionan información para el apoyo en la toma de decisiones donde los requisitos de información pueden identificarse. Las decisiones respaldadas por estos sistemas frecuentemente se repiten.
Sistema para el soporte de decisiones (DSS).	Ayudan a los directivos cuando tienen que tomar <i>decisiones no muy estructuradas</i> . Una decisión se considera no estructurada si no existen procedimientos claros para tomarla y tampoco es posible identificar, con anticipación, todos los factores que deben considerarse en la decisión.	Ayudan a los gerentes en la toma de decisiones únicas y no reiteradas. Los DSS deben tener una flexibilidad mayor que la de los demás sistemas de información.
Sistemas Expertos	Son programas de computadoras que contienen el conocimiento y la capacidad necesaria para razonar información a nivel de personas expertas en el tema.	Realizan procesos mecanizados y para los cuales el hombre ya tiene una respuesta concreta.

Otra clasificación de los SI:

- Sistema
-
- s en tiempo real. Controlan un ambiente recibiendo datos, procesándolos y devolviéndolos con la suficiente rapidez como para influir en dicho ambiente en ese momento.
- Sistemas en línea. Aceptan material de entradas directamente del área donde se creó. Son sistemas en los que el material de salida, o el resultado de la computación se devuelve directamente en donde es requerido.

Características del software:

- El software es desarrollado, no se fabrica en sentido clásico.
- Los costes del software se concentran en la ingeniería.
- El software no se “estropea”. Durante su vida el software sufre cambios.
- La mayoría del software se construye a medida, en vez de ensamblando componentes existentes.

Aplicaciones del software: Puede aplicarse en cualquier situación en la que se haya definido previamente un conjunto específico de pasos procedimentales (es decir, un algoritmo).

Tipo de software	Características
Software de sistemas	Es una colección de programas escritos para servir a otros programas, por ejemplo: compiladores, editores y utilidades de gestión de archivos.
Software de tiempo real	El software que mide, analiza, controla sucesos del mundo real conforme ocurren. Debe responder dentro de ligaduras estrictas de tiempo.
Software de gestión	Las aplicaciones en esta área reestructuran los datos existentes en orden a facilitar las operaciones comerciales o gestionar la toma de decisiones.
Software de ingeniería y científico	Se ha caracterizado por los algoritmos de "manejo de números". Las aplicaciones van desde la astronomía a la vulcanología.
Software de empujado (Firmware)	Reside en memoria de sólo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumidores. Puede ejecutar funciones muy limitadas y esotéricas, por ejemplo: el control de las teclas de un horno de microondas.
Software de computadoras personales	El procesamiento de palabras, las hojas de cálculo, gestión de bases de datos, aplicaciones financieras son sólo unos cuantos de los cientos de aplicaciones.
Software de inteligencia artificial	Hace uso de algoritmos no numéricos para resolver problemas complejos que no son adecuados para el cálculo o análisis directo.

3.3.1 Crisis del software

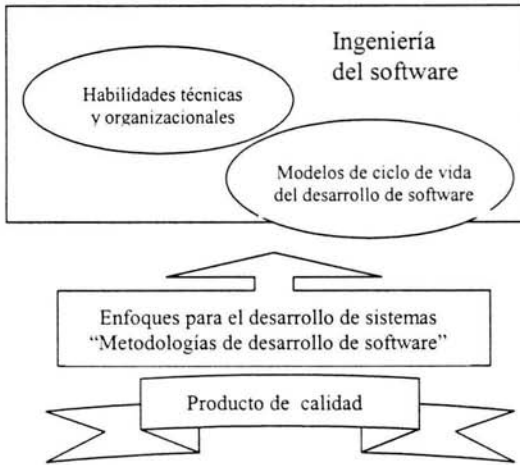
La rápida expansión de la informática ha llevado a la escritura de millones de líneas de código, antes de que se empezaran a plantear de manera seria, metodologías para la construcción y el diseño de sistemas software, así como métodos para resolver los problemas de mantenimiento, fiabilidad, etc. Esta expansión sin control tuvo como consecuencia lógica la llamada *Crisis del Software*.

El punto básico de esta crisis es que el software es mucho más difícil de construir de lo que indica la intuición. Los síntomas de esta crisis son:

- Los sistemas no responden a las expectativas del usuario.
- Los programas fallan demasiado.
- Los costos son muy superiores a lo esperado.
- La modificación de software es una tarea compleja costosa y propensa a errores.
- El software se suele entregar tarde y con menos prestaciones de las ofertadas.
- Es difícil cambiar un programa de su entorno hardware.
- Los esfuerzos que se hacen para el desarrollo de software no suelen hacer un aprovechamiento óptimo de los recursos accesibles.

El primer reconocimiento público de la existencia de esta crisis tuvo lugar en una conferencia organizada en 1968, por la Comisión de Ciencias de la OTAN que convocó en Garmisch Alemania a un grupo de unas cincuenta personas, formado por programadores de primera categoría, profesores de informática y empresarios del sector. El objetivo de esta conferencia era trazar el rumbo que permitiera salir de la crisis del software. Es así como surge la ingeniería de software.

3.3.2 Ingeniería de software



“Ingeniería del Software es el estudio de los principios y metodologías para desarrollo y mantenimiento de sistemas de software”⁹.

La ingeniería de software es una metodología multicapa, sus capas son:

- Los cimientos están orientados hacia un *enfoque de calidad*.
- El fundamento de la ingeniería de software es la capa *proceso* que es la unión que mantiene juntas las capas de tecnología y permite un desarrollo racional y oportuno de la ingeniería de software.
- Los *métodos* de la ingeniería de software indican cómo construir técnicamente el software. Abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento; dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas.
- Las *herramientas* de la ingeniería de software proporcionan un soporte automático o semiautomático para el proceso y para los métodos.

Un proceso de software se establece definiendo un pequeño número de actividades del marco de trabajo que son aplicables a todos los proyectos del software, con independencia de su tamaño o complejidad, cada uno es una colección de tareas de ingeniería del software: hitos de proyectos, posibles entregas y productos de trabajo de software, y puntos de garantía de calidad, que permiten que las actividades del marco de trabajo se adapten a las características del proyecto del software y a los requisitos del equipo del proyecto.

⁹ Zelkovitz, M. V., Shaw, A. C. y Gannon, J. D. Loc. cit.

Se selecciona un *modelo de proceso para la ingeniería del software* según la naturaleza del proyecto y de la aplicación, los métodos y las herramientas a utilizarse, y los controles y entregas que se requieren.

3.3.3 Modelos de proceso de software

Para resolver los problemas reales de una industria, un ingeniero de software ó un equipo de ingenieros deben incorporar una estrategia de desarrollo que acompañe al *proceso, métodos, capas de herramientas y tres fases genéricas*:

- Fase de *definición* se centra sobre el qué.
- Fase de *desarrollo* se centra sobre el cómo.
- Fase de *mantenimiento* se centra en el cambio que va asociado a la corrección de errores.

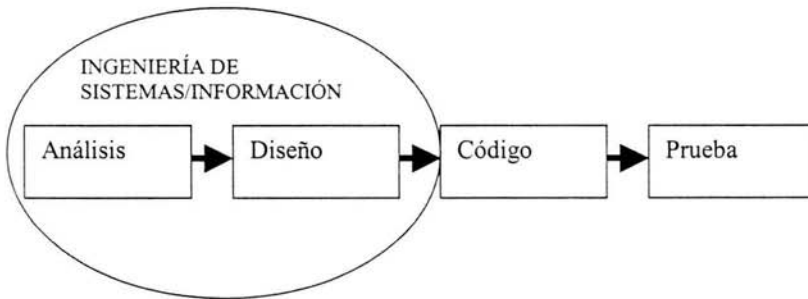
Esta estrategia a menudo se llama *modelo de proceso o paradigma de ingeniería de software*. Todo el desarrollo del software se puede caracterizar como un bucle de resolución de problemas en el que se encuentran cuatro etapas distintas:

- Status quo: “representa el estado actual de sucesos”.
- Definición de problemas: identifica el problema específico a resolverse.
- Desarrollo técnico: resuelve el problema con aplicación de alguna tecnología.
- Integración de soluciones: ofrece los resultados a los que solicitan la solución en primer lugar.

El modelo lineal secuencial: “Ciclo de vida básico” o “Modelo en cascada”.

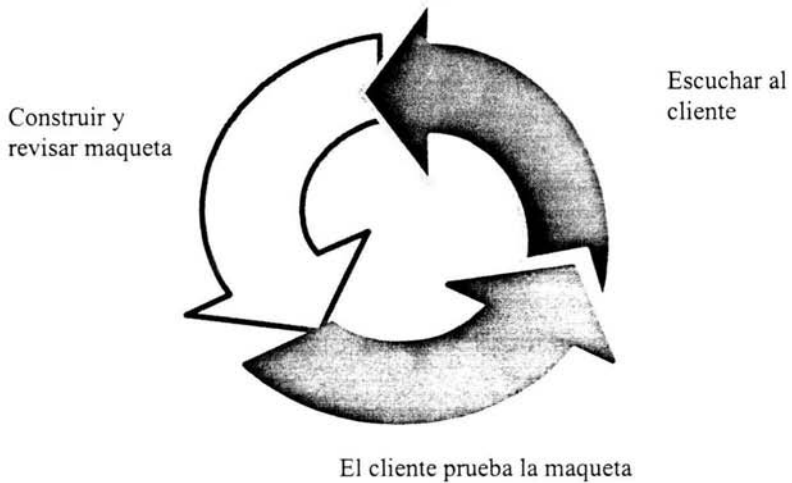
- El análisis de los requisitos del software es el proceso de reunión de requisitos se intensifica y se centra especialmente en el software.
- El diseño se centra en cuatro atributos distintos de un programa: Estructura de datos, arquitectura del software, representaciones de interfaz y detalle procedimental (algoritmo).
- En la generación de código, el diseño se debe traducir en una forma legible por la máquina.
- Las pruebas se centran en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales, es decir, la realización de las pruebas para la detención de errores.
- El mantenimiento vuelve a aplicar cada una de las fases precedentes a un programa ya existente y no a uno nuevo.

El ciclo de vida clásico sigue siendo el modelo de proceso más extensamente utilizado por la ingeniería de software.



Modelo de construcción de prototipos

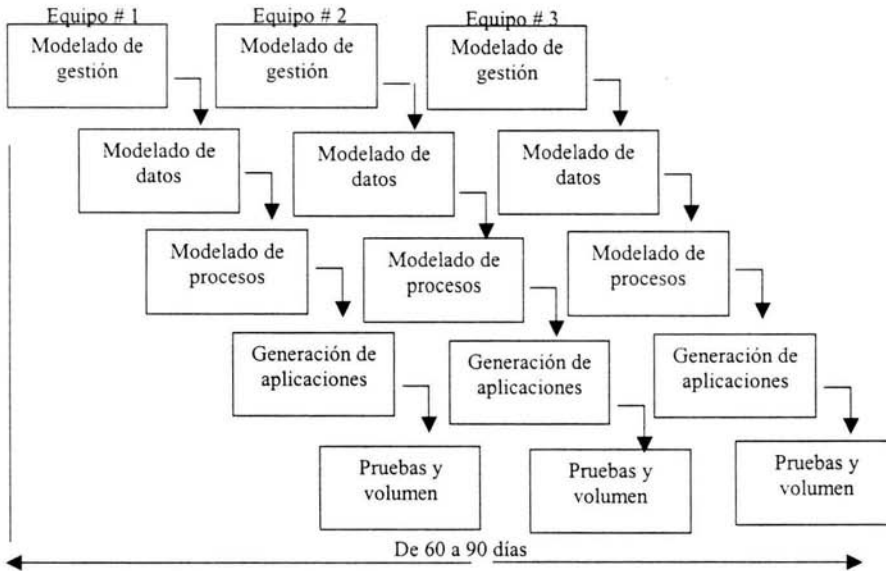
El paradigma de construcción de prototipos comienza con la recolección de requisitos. El desarrollador y el cliente encuentran y definen los objetivos globales para el software, identifican los requisitos conocidos, y las áreas del esquema en donde es obligatoria más definición. Entonces aparece un diseño rápido, se centra en una representación de estos aspectos del software que serán visibles para el usuario/cliente. El prototipo lo evalúa el cliente/usuario y lo utiliza para refinar los requisitos del software a desarrollar.



Modelo DRA (Rapid Application Development)

Es una adaptación a “alta velocidad” del modelo lineal secuencial en el que se logra el desarrollo rápido utilizando un enfoque de construcción basado en componentes. Si se comprenden bien los requisitos y se limita el ámbito del proyecto, el proceso DRA permite al equipo de desarrollo crear un sistema completamente funcional” dentro de periodos cortos de tiempo. Cuando se utiliza principalmente para aplicaciones de sistemas de información, el enfoque DRA comprende las siguientes fases:

- Modelado de gestión.
- Modelado de datos.
- Modelado de procesos.
- Generación de aplicaciones.
- pruebas y entrega.

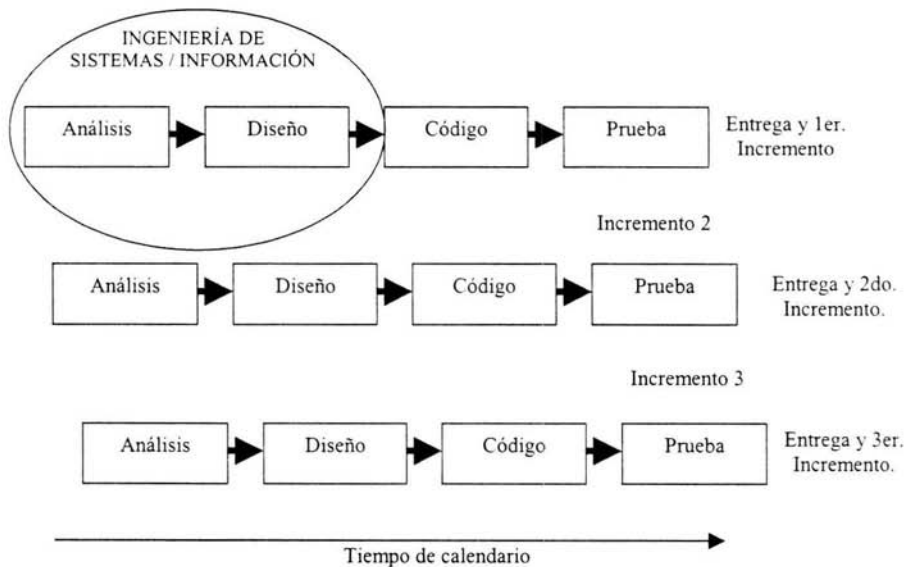


Modelos de procesos evolutivos

Los modelos evolutivos son iterativos, se caracterizan por la forma en que permiten a los ingenieros del software desarrollar versiones cada vez más completas de software.

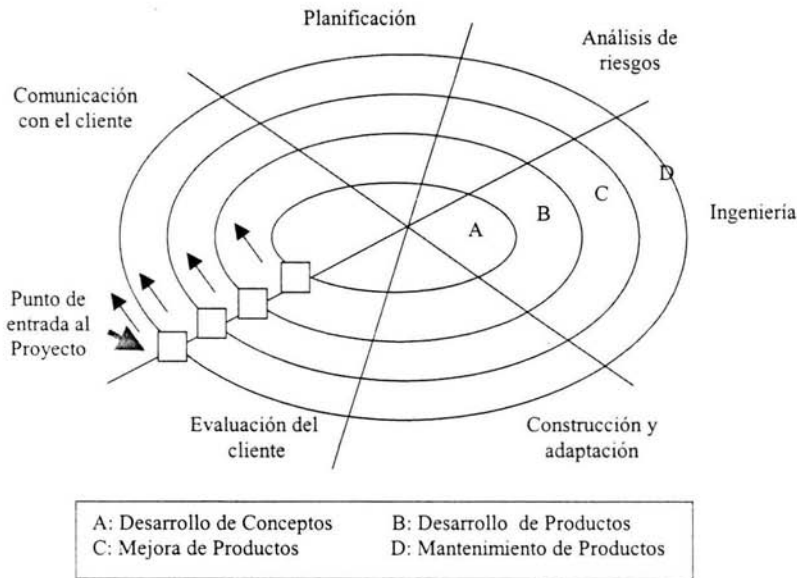
➤ Modelo incremental

Cuando se utiliza un modelo incremental, el primer incremento a menudo es un producto esencial (núcleo). Es decir, se afrontan requisitos básicos pero muchas funciones suplementarias quedan sin extraer. El cliente utiliza el producto central (o sufre la revisión detallada). Como un resultado de utilización y/o de evaluación, se desarrolla un plan para el incremento siguiente. El plan afronta la modificación del producto central a fin de cumplir mejor las necesidades del cliente y la entrega de funciones, y características adicionales.



➤ Modelo en espiral

Utiliza la construcción de prototipos como mecanismos de reducción de riesgos, pero lo que es más importante, permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto. Mantiene el enfoque sistemático de los pasos sugeridos por el ciclo de vida clásico, pero incorpora al marco de trabajo interactivo que refleja de forma más realista el mundo real.

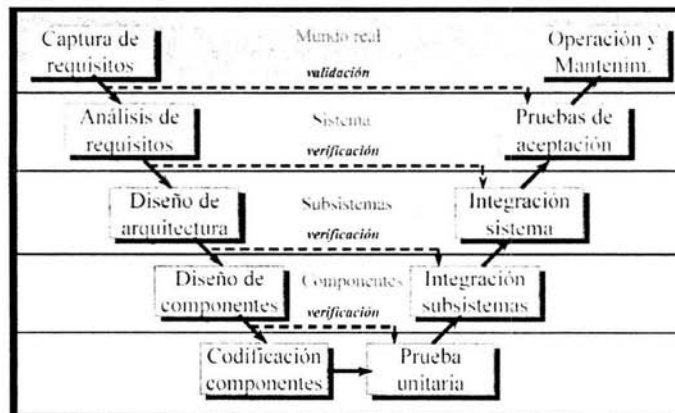


Modelo en V

El modelo en V es una variación del modelo en cascada que demuestra cómo se relacionan las actividades de prueba con las de análisis y diseño. Como se observa en la figura la codificación forma la punta de la V. Con el análisis y el diseño a la izquierda y la prueba y el mantenimiento a la derecha. La prueba unitaria y de integración se ocupa de la exactitud de los programas.

El modelo en V sugiere que la prueba unitaria y de integración también sea utilizada para verificar el diseño del programa. Es decir, durante la prueba unitaria y de integración, los codificadores y los miembros de equipo de prueba deban asegurar que todos los aspectos del diseño del programa se han implementado correctamente en el código. De igual modo, la prueba del sistema debe verificar el diseño del sistema, asegurando que todos los aspectos del diseño del sistema están correctamente implementados. La prueba de aceptación, que es dirigida por el cliente, en lugar del desarrollador, valida los requerimientos asociando un paso de prueba con cada elemento de la especificación; este tipo de prueba sirve para comprobar si todos los requerimientos se han implementado por completo, antes de que el sistema sea aceptado y pagado.

La vinculación entre los lados derecho e izquierdo del modelo V implica que, si se encuentran en problemas durante la verificación y la validación, entonces el lado izquierdo de la V puede ser ejecutado nuevamente, para solucionar el problema y mejorar los requerimientos, el diseño y el código ante de retomar los pasos de prueba sobre el lado derecho.



Modelo de métodos formales

Los métodos formales pueden describirse como prácticas de ingenierías rigurosas, basadas en fundamentos matemáticos. Pueden proporcionar un modo de expresar diseños de sistemas de manera concisa y sin ambigüedades, y de una forma en el que el comportamiento de los sistemas puede ser examinado a través de la manipulación lógica formal.

La desventaja que presenta es que el conocimiento y experiencia debe ser altamente especializado.

Técnicas de cuarta generación

Abarca un amplio espectro de herramientas de software que tienen algo en común, todas facilitan al ingeniero del software la especificación de algunas características del software de alto nivel, luego, la herramienta genera automáticamente el código fuente basándose en la especificación de técnico.

Las herramientas tales como CASE y bibliotecas de código o de clases están en un punto intermedio entre los lenguajes de alto nivel y las herramientas de codificación en su capacidad para mejorar la productividad; se dirigen a la raíz del problema, pero la mayoría de ellas no lo hacen con mucho énfasis.

Tecnología de procesos

Se han desarrollado estas herramientas para ayudar a las organizaciones de software a analizar los procesos actuales, organizar tareas de trabajo, controlar y supervisar el progreso y gestionar la calidad técnica.

Así los modelos de ciclo de vida existentes y todo lo referente a la ingeniería del software proporcionan una sana base para el origen de nuevos métodos de desarrollo de software. Dadas esas limitaciones, la adopción de un modelo de ciclo de vida apropiado es un paso importante hacia el mejoramiento de la calidad. Es el primer nivel para tener bajo control el proceso de desarrollo de software. Una vez que esto se haya logrado, se abrirá el camino para estudiar refinar y automatizar este proceso.

3.4 Monografía de las metodologías de desarrollo de software

En ésta sección presentamos una recopilación de información de las metodologías de desarrollo de software más utilizadas actualmente por los desarrolladores de software.

3.4.1 Metodologías estructuradas



En las metodologías de análisis y diseño estructurado, se examinan los sistemas desde el punto de vista de las funciones o tareas que se deben realizar, tareas que se van descomponiendo sucesivamente en otras más pequeñas y que forman los bloques o modelos de las aplicaciones.

Este tipo de metodologías produce una división entre los dos elementos de un sistema:

- Funciones, que llevan a cabo los programas.
 - Datos, que se almacenan en archivos o bases de datos.
- Metodologías orientadas a datos jerárquicos: Fundadas sobre el modelo básico entrada/proceso/salida. Se enfocan en el proceso (JACKSON) (CAMERON) (WARNIER).
- El proceso de diseño define estructuras de datos de entrada/salida, hace una estructura jerárquica de programa y ordena la lógica procedimental.
 - Se definen las estructuras de datos, el diseño lógico debe preceder al físico.

El desarrollo de sistemas estructurados de datos, también denominado **metodología Warnier-Orr**, evolucionó desde el análisis del dominio de información realizado por Warnier¹⁰, con una notación para representar la jerarquía de información mediante las tres construcciones: secuencia, selección y repetición, y demostró que la estructura del software se podría derivar directamente de la estructura de datos.

¹⁰ Warnier, J.D. , *Logical Construction of Programs* , Van Nostrand Reinhold, 1981

Ken Orr¹¹ amplió el trabajo de Warnier en el **Desarrollo de Sistemas Estructurados de Datos (DSED)**, que estudia el flujo de información y características funcionales así como la jerarquía de datos.

El Desarrollo de Sistemas Jackson (DSJ): Evolucionó fuera del trabajo realizado por Jackson¹² sobre el análisis del dominio de la información y su relación con el diseño de programas y sistemas, se centra en los modelos del dominio de información del “mundo real”.

Para realizar el DSJ el analista aplica los siguientes pasos:

- Entidades y acciones. Se identifican entidades (personas, objetos u organizaciones que un sistema necesita para producir o utilizar información) y acciones (los sucesos que ocurren en el mundo real que afectan a las entidades).
- Estructura de entidades. Acciones que afectan a cada entidad se ordenan por tiempo y se representan con diagramas Jackson (notación tipo árbol).
- Modelado inicial. Las entidades y acciones se representan como un modelo de proceso; se definen las conexiones entre el modelo y el mundo real.
- Planificación del sistema. Se evalúan y se especifican las características.
- Implementación. El hardware y software se especifican como un diseño.
-
- Metodologías orientadas a procesos: Fundadas sobre el modelo básico entrada/proceso/salida. Se enfoca en el proceso (DE MARCO) (GANE & SARSON) (YOURDON).
- **Yourdon:** Es usado ampliamente en USA y en Europa. Es un conjunto de notaciones y técnicas con líneas guía para su uso efectivo para el análisis y diseño estructurado, sugiere modelos diferentes y no impone su uso por lo que es ampliamente flexible.

Principios:

- Modelo de viabilidad de estudio
- Modelo del sistema.
- Aplicaciones de tiempo real y procesamiento de datos.
- Cubre las etapas de viabilidad, análisis y diseño del ciclo de vida convencional.

Yourdon describe técnicas para la realización de análisis estructurado de sistemas basado principalmente en los siguientes conceptos:

- Diagramas de flujo de datos para la representación de procesos.

¹¹ Orr K. T. , *Structured Requirements Definition*, Ken Orr & Associates, Inc., Topeka, KS.1981

¹² Jackson M.A., *System Development*, Prentice Hall, 1983.

- Diagramas de transición de estados para la representación estructurada de las funciones a realizar en los procesos.
- Modelo Entidad/Relación para la representación conceptual de los datos.
- Diccionario de datos como base o soporte de información del sistema.
- Diagramas o mapas de estructura para la representación modular de los procesos y las variables intercambiadas entre ellos.
- Especificaciones de programas basadas en lenguaje estructurado y tablas de decisión.

Structured Analysis and Design Technique (SADT). La técnica de análisis y diseño estructurado se ha utilizado mucho como una notación para definición de sistemas, representaciones de procesos, análisis de requisitos del software y diseño del sistema/software.

SADT se compone de procedimientos que permiten al analista descomponer las funciones del software (o sistema); una notación gráfica, el actigrama y datagrama SADT que comunica las relaciones de información (datos y control) y función dentro del software, y las directrices de control de proyectos para aplicar la metodología.

La metodología SADT es acompañada de herramientas automáticas para poder soportar procedimientos de análisis y de disposiciones organizativas de cómo utilizar correctamente las herramientas. Se especifican las revisiones y los hitos, permitiendo la validación de la comunicación desarrollador-cliente.

- Metodologías orientadas a datos no jerárquicos: Los datos son el corazón del sistema de información. El modelo está formado por el conjunto de entidades básicas y las interrelaciones entre ellas (MARTIN & FINKELSTEIN).

Ingeniería de la información: Consta de cuatro etapas:

- Planificación: Construir una arquitectura de la información y una estrategia que soporte los objetivos de la organización.
 - Análisis: Comprender las áreas del negocio y determinar los requisitos del sistema.
 - Diseño: Establecer el comportamiento del sistema deseado por el usuario y que sea alcanzable por la tecnología.
 - Construcción: Construir sistemas que cumplan con los tres niveles anteriores.
- Mixtas: Metodologías que combinan los enfoques de orientadas a procesos y orientadas a datos.

Structured Systems Analysis and Design (SSADM). Nace en el Reino Unido a petición del gobierno inglés con la intención de ser un sistema de desarrollo de aplicaciones informáticas para toda la administración pública de Gran Bretaña. Es aceptada en 1981 en que se presenta la primera versión.

SSADM resulta en un modelo de proceso y modelo de datos. Los modelos son construidos en paralelo y uno es usado para verificar la consistencia del otro. Esta construcción verifica de manera consistente que el enfoque es la mayor ventaja que ofrece SSADM. El modelamiento es tratado de manera más completa por SSADM que por otros métodos. SSADM no soporta el diseño de sistemas en tiempo real.

La metodología consiste en una estructuración de los pasos a seguir en el desarrollo de un proyecto informático en las fases iniciales del ciclo de vida del mismo y en la descripción de unas técnicas y formalismos sobre las que se basan los trabajos a realizar en cada fase.

SSADM crea y perfecciona el modelo del sistema mediante cuatro etapas:

- Modelo físico actual
- Modelo lógico actual
- Modelo lógico requerido
- Modelo físico requerido

La meta de la fase SR (System Requirement) deberá ser la construcción de un modelo lógico requerido. El modelo físico requerido incorpora las consideraciones de implementación y su construcción debe ser deferida a la fase AD (Analysis and Design).

El enfoque SSADM asume que un sistema está siendo reemplazado. El modelo físico actual describe la forma presente de hacer las cosas. Esto debe ser analizado para crear el modelo lógico actual, y después combinado con 'la lista de problemas y requerimientos' de los usuarios para construir el modelo lógico requerido.

Este concepto de evolución puede ser aplicado muy a menudo. La mayoría de los sistemas tienen un claro ancestro, y puede aprenderse mucho estudiando a sus predecesores. Esto evita que la gente 'reinvente la rueda'. El URD (User Requirements Definition) siempre debe describir o hacer referencia a sistemas similares, de modo que el desarrollador pueda entender mejor el contexto de los requerimientos del usuario. Los estándares de ingeniería de software exigen explícitamente que la relación con proyectos predecesores esté documentada en el SRD (System Requirement Definition).

Si un sistema está siendo reemplazado y se requiere un sistema de procesamiento de datos, entonces se recomienda el enfoque SSADM.

Métrica versión 3: Tiene un enfoque orientado al proceso, ya que la tendencia general de los estándares se encamina en este sentido. Métrica versión 3 cubre el proceso de desarrollo y el proceso de mantenimiento de SI.

La metodología descompone cada uno de los procesos en actividades, y éstas a su vez en tareas. Para cada tarea se describe su contenido haciendo referencia a sus principales acciones, productos, técnicas, prácticas y participantes.

El orden asignado a las actividades no debe interpretarse como secuencia en su realización, ya que éstas pueden realizarse en orden diferente a su numeración o bien en paralelo, como se muestra en los gráficos de cada proceso. Sin embargo, no se dará por acabado un proceso hasta no haber finalizado todas las actividades del mismo determinadas al inicio del proyecto.

Así los procesos de la estructura principal de métrica versión 3 son los siguientes:

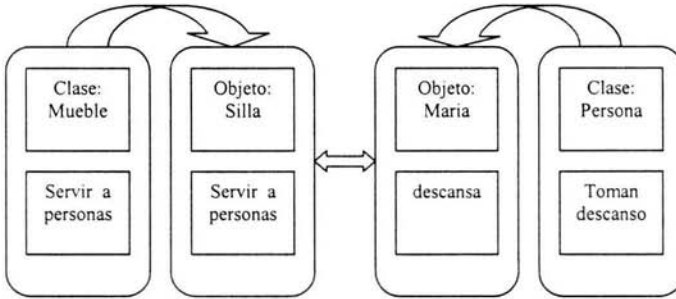
- Planificación de sistemas de información.
- Desarrollo de sistemas de información.
- Mantenimiento de sistemas de información.

La existencia de tecnología de reciente aparición, permite disponer de sistemas que apoyan la toma de decisiones a partir de grandes volúmenes de información procedentes de los sistemas de gestión e integrados en una plataforma corporativa. Métrica versión 3 ayuda en la planificación de sistemas de información facilitando una visión general necesaria para posibilitar dicha integración y un modelo de información global de la organización.

En cuanto al proceso de desarrollo de sistemas de información, para facilitar la comprensión y dada su amplitud y complejidad se ha subdividido en cinco procesos:

- Estudio de viabilidad del sistema (EVS).
- Análisis del sistema de información (ASI).
- Diseño del sistema de información (DSI).
- Construcción del sistema de información (CSI).
- Implantación y aceptación del sistema (IAS).

3.4.2 Metodologías orientadas a objetos (OO)



En las metodologías de orientación a objetos, el aspecto de “modelado” del sistema cobra demasiada importancia, examinando el dominio del problema como un conjunto de objetos que interactúan entre sí. Además da un enfoque unificado de los dos elementos de un sistema, como son: las funciones y los datos, que se unen en los objetos.

La OO se basa alrededor de ideas de información oculta, inherencia, encapsulación de datos y tipos de información abstracta. El enfoque visualiza todos los recursos como son la información y el sistema mismo, a manera de objetos.

Consta de cuatro etapas:

- Identificar objetos y atributos.
- Identificar operaciones en los objetos.
- Establecer las interfaces.
- Implantar las operaciones.

Orientado a objetos significa que el sistema se organiza como una colección de objetos que interactúan entre sí y que contienen tanto estructuras de datos como un comportamiento. Esto se opone a la programación convencional, en la cual las estructuras de datos y el comportamiento solamente están relacionadas de forma débil, ya que estos se enfocan principalmente a las funciones.

Los objetos son las cosas físicas y conceptuales que encontramos en el universo alrededor de nosotros. Hardware, software, documentos, seres humanos, los conceptos son todos los ejemplos de los objetos.

Características de los Objetos:

Identidad: Los datos están cuantificados en entidades discretas y distinguibles denominadas objetos. Ejemplo una televisión, una bicicleta, un árbol. Los objetos pueden ser concretos, como un archivo en un sistema de archivos, o bien conceptuales como la

política de planificación en un sistema operativo con multiprocesos. Cada objeto posee su propia identidad inherente.

Clasificación: Significa que los objetos con la misma estructura de datos (atributos) y comportamiento (operaciones) se reúnen para formar una **clase**. La selección de clases es arbitraria y depende de la aplicación.

Polimorfismo: Significa que una misma operación puede comportarse de modos distintos en distintas clases. La operación mover por ejemplo se puede comportar de modo distinto en las clases Ventana y Pieza de ajedrez. Una operación es una acción o una transformación que se lleva a cabo o que se aplica a un objeto. Justificar a la derecha, visualizar y mover son ejemplos de operaciones.

Herencia: Es compartir atributos y operaciones entre clases tomando como base una relación jerárquica. En términos generales se puede definir una clase que después se irá refinando sucesivamente para producir subclases. Todas las subclases poseen o heredan, todas y cada una de las propiedades de su superclase y añaden, además, sus propiedades exclusivas. No es necesario repetir las propiedades de las superclases en cada subclase.

La popularidad de las tecnologías de objetos ha generado docenas de métodos de análisis y diseño orientado a objetos. Cada uno de ellos introduce un proceso para el análisis de un producto o sistema, un conjunto de modelos que evoluciona fuera del proceso, expande un conjunto de representaciones de diseño, y una notación que posibilita al ingeniero del software crear y diseñar cada modelo de una manera consistente.

La OO propugna un enfoque unificado, existen dos enfoques:

1.- Revolucionarias o puras (OOD-BOOCH) (WIRFS & BROCK).

El método de Booch [Booch 1994]¹³: La metodología Booch se enfoca principalmente al diseño de estado de un proyecto. Booch describe una serie de propiedades generales de los sistemas complejos bien estructurados. En el análisis y diseño OO, el dominio del problema se modela a partir una estructura lógica del sistema y una estructura física. Y en cada una se modela la semántica.

La metodología Booch define diferentes modelos para la descripción de un sistema. El modelo lógico (dominio del problema) se representa en la estructura clase-objeto. En el diagrama de clase, se construye la arquitectura y el modelo estático.

¹³ Booch, G., Object Oriented Analysis and Design, 1994.

Procesos "macro":

- Establecer los requerimientos del núcleo (conceptualización).
- Desarrollar un modelo del comportamiento deseado (análisis).
- Crear la arquitectura (diseño).
- Evolucionar la implementación (evolución).
- Administrar la evolución posterior a la entrega (mantenimiento).

Procesos "micro":

- Identificar las clases y objetos a cierto nivel de abstracción.
- Identificar la semántica de estas clases y objetos.
- Identificar las relaciones entre las clases y objetos.
- Especificar la interfaz y después la implementación de las clases y objetos.

La metodología Booch cubre las fases de análisis y diseño de un sistema OO. Esta metodología es criticada por el uso de muchos símbolos diferentes, ya que define muchos símbolos que documentar casi para cada decisión de diseño. Si se trabaja con esta metodología, uno se perca de que nunca se usan todos estos símbolos y diagramas. Se comienza con diagramas clase-objeto en la fase de análisis y se depuran en varios pasos. Únicamente cuando se está listo para generar código, se agregan algunos símbolos de diseño. Y está es la parte en que la metodología Booch es fuerte, realmente es posible documentar el código orientado a objetos.

El método de Jacobson [Jacobson 1992]¹⁴: También llamado ISOO (Ingeniería del Software Orientado a Objetos), el método Jacobson es una versión simplificada de Objectory, un método patentado, también desarrollado por Jacobson. Este método se diferencia de los otros por la importancia que da al caso de uso¹⁵. En términos de ciclo vital, ISOO recomienda etapas de análisis, construcción y pruebas. La construcción a su vez se descompone en diseño e implementación. La obtención de requerimientos precede al análisis, y se encuentra fuera de ISOO.

A grandes rasgos el modelo consta de pasos generales:

- El modelo de requerimientos: El objetivo es la captura de requerimientos.
 - El modelo de análisis: El objetivo es dar al sistema una estructura de objetos robusta y flexible a los cambios.
 - Modelo de diseño: Tiene como objetivo adoptar y refinar la estructura de objetos en el ambiente actual de implementación.
 - El modelo de implementación: Tiene como objetivo implementar el sistema.
 - El modelo de prueba: Su objetivo es verificar el sistema.
- 2.- Sintetistas o evolutivas (OMT-RUMBAUGH) (MARTIN & ODELL).

¹⁴ Jacobson, I., *Object-Oriented Software Engineering*, Addison Wesley, 1992

¹⁵ Es una descripción o escenario que describe como el usuario interactúa con el producto o sistema.

El método de Rumbaugh [Rumbaugh 1991]¹⁶: James Rumbaugh y sus colegas desarrollaron la Técnica de Modelado de Objetos (OMT) para el análisis, y diseño del nivel de objetos. La fase de análisis comienza con la declaración del problema que incluye, una lista de objetivos (metas) y conceptos claves definitivos definidos para el dominio del problema a resolver. La declaración del problema se expande después en tres modelos:

- El modelo de objeto (una representación de objetos, clases, jerarquías, y relaciones).
- El modelo dinámico (una representación del comportamiento del sistema y los objetos: como eventos, estados y transiciones).
- El modelo funcional (una representación a alto nivel del flujo de información a través del sistema similar al DFD).

La fase de análisis genera diagramas del modelo de objetos, diagramas de estado, diagramas de eventos de flujo y diagramas de flujos de datos. Es entonces cuando se tiene completa la fase de análisis.

Después de la fase de análisis, se sigue con la fase de diseño de sistema. Aquí se define la arquitectura completa del sistema. Primero el sistema se organiza en subsistemas que están asignando a ciertos procesos y tareas, tomando en cuenta la colaboración y concurrencia entre ellos. Luego, el almacenamiento persistente de datos se establece por medio de una estrategia de manejo de información global compartida. Después, se examinan las situaciones límite para ayudar a establecer las prioridades de negociación.

La fase de diseño de objetos viene después de la fase de diseño del sistema. Aquí se establece el plan de implementación. Se definen las clases de objetos, así como sus algoritmos, poniendo especial atención con la optimización y persistencia de datos. Se definen cuestiones de herencia, asociaciones, agregaciones y valores por omisión.

➤ Segunda generación: Proceso Unificado de Desarrollo de Software. (RUP) (Método Unificado BOOCH & RUMBAUGH).

El Proceso Unificado es un proceso de desarrollo de software¹⁷. Sin embargo es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software en diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes tipos de aptitud y diferentes tamaños de proyecto.

El Proceso Unificado utiliza el *Lenguaje Unificado de Modelado* (Unified Modeling Language, UML) para preparar todos los esquemas de un sistema software. Además guía a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado mientras se balancean los requerimientos del negocio, el tiempo al mercado y los riesgos

¹⁶ Rumbaugh et al., *Object Oriented Modeling and Design*, Prentice Hall, 1991.

¹⁷ Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos necesarios de un usuario en un sistema software.

del proyecto. El proceso describe los diversos pasos involucrados en la captura de los requerimientos y en el establecimiento de una guía arquitectónica lo más pronto, para diseñar y probar el sistema.

El Proceso Unificado es soportado por herramientas que automatizan entre otras cosas, el modelado visual, la administración de cambios y las pruebas. El Proceso Unificado es un proceso porque "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar cierto objetivo, en este caso el desarrollo de software" [Jacobson 1998]¹⁸.

Según [Booch 1998]¹⁹, los conceptos clave del Proceso Unificado son:

- Fase e iteraciones: ¿Cuándo se hace?
- Flujos de trabajo de procesos (actividades y pasos): ¿Qué se está haciendo?
- Artefactos (modelos, reportes, documentos): ¿Qué se produjo?
- Trabajador (un arquitecto): ¿Quién lo hace?

El Proceso Unificado ha adoptado un enfoque que se caracteriza por:

- Interacción continua con el usuario desde un inicio.
- Mitigación de riesgos antes de que ocurran.
- Liberaciones frecuentes.
- Aseguramiento de la calidad.
- Involucramiento del equipo en todas las decisiones del proyecto.
- Anticiparse al cambio de requerimientos.

Las características primordiales del Proceso Unificado son:

- Iterativo e incremental.
- Centrado en la arquitectura.
- Guiado por casos de uso.
- Confrontación de riesgos.

Las fases del ciclo de vida del software son [Booch 1998]²⁰:

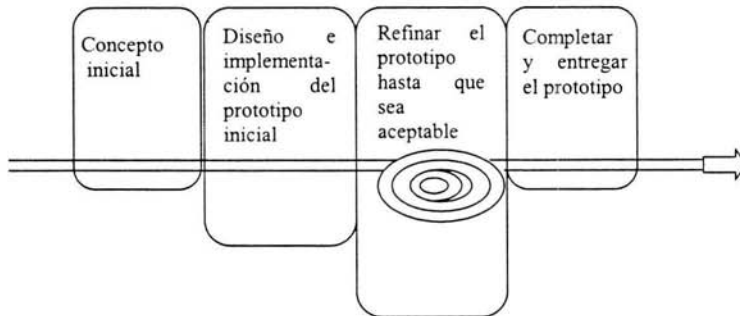
- Concepción: Es definir el alcance del proyecto y definir el caso de uso.
- Elaboración: Es proyectar un plan, definir características y cimentar arquitectura.
- Construcción: Es crear el producto.
- Transición: Es transferir el producto a sus usuarios.

¹⁸ Jacobson, I. 1998. "Applying UML in The Unified Process".

¹⁹ Booch G. 1998. Software Architecture and the UML.

²⁰ Booch G. Loc. cit.

3.4.3 Metodologías de prototipo



La metodología de prototipo acelera el proceso de desarrollo al presentar al usuario una versión del producto final, al inicio del proceso de desarrollo. A partir de ahí, el esfuerzo de desarrollo se concentra en refinar el producto acordado. El principal beneficio que tiene es la oportunidad de clarificar (o bien modificar los requerimientos del usuario) el proceso de prototipo. Es interactivo y dirigido por el usuario, además es flexible y es una antitesis de las actuales metodologías estructuradas. El nivel de injerencia del usuario durante la creación del prototipo debe ser muy alto, aunque se puede considerar como el peligro principal de este enfoque, ya que se debe saber cuando dejar de refinar y cuando consentir con el sistema final.

Tipos de Prototipos:

- Prototipos de viabilidad: Se utilizan para probar la viabilidad de una tecnología específica aplicable a un sistema de información.
- Prototipos de necesidades o prototipos de descubrimiento: Durante la elaboración de diseños de necesidades, el analista puede pintar pantallas o informes de muestra, y solicitar las opiniones del usuario con respecto a su contenido (pero no su formato).
- Prototipos de diseño o prototipos de comportamiento: Se utilizan para simular el diseño del sistema de información final.
- Prototipos de implantación o prototipos de producción: Constituyen una extensión de los prototipos de diseño donde el prototipo evoluciona directamente hacia el sistema de producción.

Existen tres ventajas relevantes en el uso de prototipo:

- Pronta modificación del sistema en su desarrollo.
- Oportunidad de detener el desarrollo de un sistema que no sirve.
- Posibilidad de desarrollar otros sistemas que se ajusten mejor a las necesidades y a las expectativas del usuario.

Desventajas del uso de prototipos:

- Su administración llega a ser difícil como un proyecto de desarrollo de prototipos, dentro de un gran esfuerzo de sistemas.
- Tanto los usuarios como los analistas pueden considerar el prototipo como un sistema concluido, cuando de hecho no lo es, y nunca se plantea como un sistema final.

Para que la creación del prototipo de software sea efectiva, debe desarrollarse rápidamente para que el cliente pueda valorar los resultados y recomendar los cambios oportunos. Para poder crear prototipos rápidos hay disponibles tres clases genéricas de métodos y herramientas:

Técnicas de cuarta generación (4GT): Comprenden una amplia gama de lenguajes de consulta e informes de bases de datos, generadores de programas y aplicaciones y de otros lenguajes no procedimentales de muy alto nivel. Como las técnicas 4GT permiten al ingeniero del software generar código ejecutable rápidamente, son ideales para la generación de prototipos.

Componentes de software reutilizables. Un componente software puede ser una base de datos o un componente arquitectónico de software (p. ej.: un programa) o un componente procedimental (p. ej.: un módulo). En todos los casos se debe diseñar el componente software de manera que permita ser reutilizado sin un conocimiento detallado de su funcionamiento interno.

Especificaciones formales y entornos para prototipos. Hoy en día, los desarrolladores de los lenguajes formales están desarrollando entornos interactivos que: (1) permitan al analista crear interactivamente una especificación basada en lenguaje de un sistema o software. (2) invoquen herramientas automáticas la especificación basada en el lenguaje en código ejecutable y (3) permitan al cliente usar el código ejecutable del prototipo para refinar los requisitos formales.

3.4.4 Metodología ágil de desarrollo o evolutiva

Las metodologías ágiles son sin duda uno de los temas recientes en ingeniería de software que están acaparando gran interés. Prueba de ello es que se están haciendo un espacio destacado en la mayoría de conferencias y workshops celebrados en los últimos años. Es tal su impacto que actualmente existen 4 conferencias internacionales de alto nivel y específicas sobre el tema²¹.

Las características de los proyectos para los cuales las metodologías ágiles han sido especialmente pensadas se ajustan a un amplio rango de proyectos industriales de desarrollo de software; aquellos en los cuales los equipos de desarrollo son pequeños, con plazos reducidos, requisitos volátiles, y/o basados en nuevas tecnologías.

En febrero de 2001, tras una reunión celebrada en Utah E. U., nace el término “ágil” aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto.

Tras esta reunión se creó The Agile Alliance²², una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida es fue el Manifiesto ágil, un documento que resume la filosofía “ágil”.

Las metodologías ágiles se refieren a todos esos nuevos métodos en los que para llevar a cabo cualquier proyecto se necesite hacer énfasis en la velocidad de desarrollo. No necesariamente se identifica una herramienta o métodos específicos sino la que mejor pudiera tener profundas implicaciones en las aplicaciones más cortas. Para algunas personas, el desarrollo rápido consiste en aplicar una única herramienta o método. Para el hacker, el desarrollo rápido consiste en codificar 36 horas de un tirón. Para el ingeniero de sistemas de información es RAD (Rapid Application Development) una combinación de herramientas CASE, la participación intensiva del usuario y ventanas temporales estrictas.

Cada uno de estos métodos y herramientas va bien hasta un cierto límite, y cada uno puede contribuir a incrementar la velocidad de desarrollo. Pero para obtener todo el beneficio posible, cada uno tiene que estar coordinado dentro de una estrategia global.

²¹ XP Agile Universe: www.agileuniverse.com. Conference on eXtreme Programming and Agile Processes in Software Engineering: www.xp2004.org. Agile Development Conference (EEUU): www.agiledevelopmentconference.com. Agile Development Conference (Australia): www.softed.com/adc2003/

²² www.agilealliance.com

El movimiento del desarrollo ágil de software es una iniciativa que agrupa una serie de metodologías que se basan en la adaptabilidad ante el cambio como medio para aumentar las posibilidades de éxito de un proyecto. En general los procesos ágiles se centran en las personas; en su comunicación directa y sus habilidades en vez de en procesos muy formales.

➤ eXtreme Programming

Programación extrema (XP) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck el padre de XP, describe la filosofía de XP²³, sin cubrir los detalles técnicos y de implantación de las prácticas. Posteriormente, otras publicaciones de experiencias se han encargado de dicha tarea. A continuación presentaremos las características esenciales de XP organizadas en los tres apartados siguientes: historias de usuario, roles, proceso y prácticas.

Las historias de usuario: Son la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Los roles de acuerdo con la propuesta original de Beck son:

- Programador: El programador escribe las pruebas unitarias y produce el código del sistema.
- Cliente: Escribe las historias de usuario y las pruebas funcionales para validar su implementación.
- Encargado de pruebas (Tester): Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

²³ Beck, K.. *Extreme Programming Explained. Embrace Change*, Pearson Education, 1999. Traducido al español como: "Una explicación de la programación extrema. Aceptar el cambio", Addison Wesley, 2000.

- Encargado de seguimiento (Tracker): Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones.
- Entrenador (Coach): Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- Gestor (Big boss): Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

Proceso XP: El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

El ciclo de vida ideal de XP consiste de seis fases:

- Exploración.
- Planificación de la entrega (Release).
- Iteraciones.
- Producción.
- Mantenimiento.
- Muerte del proyecto.

Prácticas XP: La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes prácticas.

- Hay una comunicación frecuente el cliente y los programadores.
- Entregas pequeñas. Producir rápidamente versiones del sistema que sean operativas.
- Metáfora. El sistema es definido mediante una metáfora (una metáfora es una historia compartida que describe cómo debería funcionar el sistema, conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema), o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo.
- Diseño simple.
- Pruebas. La producción de código está dirigida por las pruebas unitarias.

- Refactorización (Refactoring). Es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios.
- Programación en parejas.
- Propiedad colectiva del código. Cualquier programador puede cambiar cualquier parte del código en cualquier momento.
- Integración continua. Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.
- 40 horas por semana. Se debe trabajar un máximo de 40 horas por semana.
- Cliente in-situ. El cliente tiene que estar presente y disponible todo el tiempo para el equipo.
- Estándares de programación. XP enfatiza que la comunicación de los programadores es a través del código.

➤ Crystal Methodologies

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas.

3.5 Definiciones de las metodologías de desarrollo de software

En este apartado nos dimos a la tarea de recopilar diferentes significados del concepto *metodologías de desarrollo de software*, con el objetivo de obtener una mejor comprensión del tema.

3.5.1 Definición etimológica

Diccionario Etimológico de la Lengua Española:

- **Metodología:** Del griego:
 - 1.- μέθοδος.De μέθα meta: por.
Y de οδος odos: ogo camino.
 - 2.- λογος logos: razón, discurso, tratado.Al latín: Methodologos.
Al español: Metodología.

- **Desarrollo:** Del latín desarrollo.
 - 1.- Des: Significa hacer lo contrario a.
 - 2.- A: Efecto de hacer.
 - 3.- Rollo: Del latín rotulus: cilindro.Al español: Desarrollo.

- **Software**²⁴: Del inglés: Software.
 - 1.- Soft: Blando, suave.
 - 2.- Ware: Mercancía, artículo, mercadería.Al español: Software.
 - 1.- Se mantiene como anglicismo sin alteración ortográfica.
 - 2.- Sustantivo. Masculino. Singular.
 - 3.- Es el conjunto de programas y codificaciones necesarias para hacer que el hardware ejecute las tareas ordenadas por su usuario.

²⁴ Nota: Software es un anglicismo creado como complemento al vocablo hardware utilizados ambos en computación electrónica.

3.5.2 Definición de diccionarios

Diccionario de la Real Academia Española:

- **Metodología:** (De gr. Método, y logía). F. Ciencia del método ||. 2. Conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal.
- **Desarrollo:** m. Acción y efecto de desarrollar o desarrollarse ||. 2. Combinación entre el plato y el piñón de la bicicleta, que determine la distancia que se avanza con cada pedalada ||. 3. Econ. Evolución progresiva de una economía hacia mejores niveles de vida. Nota. Se dan varios significados.
- **Software:** La Real Academia Española no define el vocablo software.

Diccionario Enciclopédico Color:

- **Desarrollar:** tr. Desenvolver una cosa que estaba arrollado, tr. y prnl. Hacer que crezca un organismo. Fig. Dar incremento a una cosa. tr. Fig. Explicar una teoría, plan etc. y llevarla a sus últimas consecuencias. Mat. Efectuar operaciones de cálculo, para cambiar la forma de una exp. Analítica. prnl. Suceder una cosa de manera o en el lugar que se expresa.
- **Software:** (Voz ing.) m. comp. Conjunto de programas que se pueden ejecutar en una computadora. En general, se distingue entre el s. de base, que incluye el sistema operativo, los compiladores y ensambladores y el conjunto de programas y rutinas o subrutinas que hacen posible su funcionamiento, y la programación agregada por el usuario.

Diccionario Conceptual de Informática y Comunicaciones:

- **Metodología:** Conjunto de herramientas y técnicas destinadas a apoyar el desarrollo ordenado de las distintas fases de elaboración de un sistema de información. Aplicable principalmente al análisis y diseño del sistema, puede cubrir también las fases de planificación, implantación y soporte. Las metodologías se componen de procedimientos normalizados de aplicación de cada tarea y fase de los proyectos, que han de sustentarse en una documentación minuciosa, completa y extendida a la totalidad del proyecto.
- **Desarrollo:** Conjunto de tareas y actividades para la elaboración de un sistema de información o de uno de sus módulos. En particular se aplica a las fases de diseño detallado y programación de un sistema.
- **Software:** Conjunto de elementos lógicos de un sistema de información, como programas, reglas y procedimientos informáticos.

3.5.3 Definición de autores

Maddison 1983

Metodología de desarrollo de software: Conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores del software.

Sommerville 2002

Un método de ingeniería de software es un enfoque estructurado para el desarrollo de software cuyo propósito es facilitar la producción de software de alta calidad de una forma costeable.

Grady Booch 1998

Una metodología es una colección de métodos aplicados a lo largo del ciclo de vida del desarrollo del software y unificados por alguna aproximación.

Un método es un proceso disciplinado para generar un conjunto de modelos que describen varios aspectos de un sistema de software en desarrollo, utilizando alguna notación bien definida

Jeffrey L. Whitten 1998

Metodología de desarrollo de software: Una metodología es una versión amplia y detallada de un ciclo de vida completo de sistemas que incluye:

- Tareas paso a paso para cada fase.
- Funciones individuales y en grupo desempeñadas en cada tarea.
- Productos resultantes y normas de calidad para cada tarea.
- Técnicas de desarrollo que se utilizaran en cada tarea.

También podemos decir que una metodología es un conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información.

3.5.4 Definición propia

Una metodología de desarrollo de software debería señalarnos “qué” pasos tomar y “cómo” llevarlos acabo, pero más importante es definir las razones del “por qué” esos pasos se deben tomar en ese orden.

Una metodología completa es algo más que una notación, un proceso y herramientas, además pueden ser, guías que nos permitan el manejo del proyecto en las tareas y entregas, políticas y procedimientos que garanticen la calidad del software, medidas y métricas, descripción de roles y entrenamiento detallados, técnicas para adaptar el método y técnicas bien definidas.

Características de las metodologías de desarrollo de software:

- Existencia de reglas predefinidas.
- Cobertura total del ciclo de vida.
- Verificaciones intermedias.
- Planificación y control.
- Comunicación efectiva.
- Utilización sobre un abanico amplio de proyectos.
- Fácil formación.
- Herramientas CASE.
- Actividades que mejoren el proceso.
- Soporte al mantenimiento.
- Soporte de la reutilización de software.

3.5.5 Definición sinónimos y antónimos

Langensheidt Diccionario Básico Sinónimos y Antónimos:

- **Sinónimo Método:** Orden, regla, norma, tratamiento, disposición, procedimiento, sistema, ordenación, proceso, manera, doctrina, planteamiento, política, táctica, criterio, costumbre.
- **Antónimo Método:** Indisciplina, desbarajuste, desconcierto, alteración, barrullo, desgobierno.

- **Sinónimo Desarrollo:** Aumento, progresar, crecimiento, desenvolvimiento, incremento, prosperidad, riqueza, perfeccionamiento, avance, auge, propagación, exposición, tratamiento, explanación, análisis.
- **Antónimo Desarrollo:** Retroceso, atraso, disminución, subdesarrollo, hundimiento, crisis.

3.6 Evolución de las metodologías de desarrollo de software

Podemos decir que para finales de la década de 1960 e inicios de la década de 1970 el análisis estructurado surge de la necesidad de buscar una forma interpretativa más rápida y eficiente, de tal manera que se pudiesen definir los requerimientos del usuario y las especificaciones funcionales del sistema ya que para esto existían documentos textuales, de este tipo:

- **Monolíticas:** Para entender el sistema había que leerse la especificación de cabo a rabo. No había posibilidad de que ni el analista ni el usuario pudiesen centrarse en una determinada parte de la especificación, sin tener que leerse el resto.
- **Redundantes:** La misma información se repartía en diferentes partes del documento. Debido a esto cualquier cambio que se hiciese en la especificación debía reflejarse en varios puntos del documento. Esta situación produce con frecuencia inconsistencia: si no se cambiaba en todos estos lugares la misma información (Ej. El mismo código) tenía definiciones distintas.
- **Ambiguas:** Al estar escritas en lenguaje natural, podían ser interpretadas de forma distinta por analistas, usuarios, diseñadores o programadores. Hay estudios que muestran que el 50% de los errores encontrados en el sistema final y el 70% del coste de la corrección de los errores surgía de este tipo de malentendidos.
- **Imposibles de mantener o modificar:** Por todas las razones anteriores, la especificación del sistema estaba totalmente obsoleta cuando finalizaba el desarrollo. En muchos casos, estaba incluso obsoleta cuando finalizaba la fase de análisis de requisitos. Debido a esto, la mayor parte del software de la época carece de una documentación fiable, incluso aunque se hubiese realizado análisis y redactado una especificación.

Por estos motivos fueron surgiendo nuevos métodos de análisis, cuyo objetivo era obtener especificaciones:

- **Gráficas:** Formadas por una colección de diagramas, acompañados de información textual detallada, que sirve de material de referencia, más que de cuerpo principal de la especificación.
- **Particionadas:** De forma que fuese posible leerse o trabajar sobre partes individuales de la especificación sin tener que leerse toda.
- **Mínimamente redundantes:** De forma que los cambios en los requisitos necesitasen reflejarse en un sólo punto de la especificación.
- **Transparentes:** De forma que fuesen tan fáciles de leer y de entender que el que las utilizase no se diese cuenta de que está mirando una representación del sistema, en lugar del sistema en sí. Los sistemas son los que son complejos, las especificaciones tienen que ser claras y sencillas.

Posteriormente, a mediados de la década de 1970 estando el análisis estructurado clásico en su apogeo aparecen una serie de dificultades que limitan al analista hacer un buen desempeño de sus actividades.

Entre estos problemas podemos mencionar:

- Distinción difusa y poca, definida entre los modelos lógicos y los modelos físicos.
- Limitación para modelar sistemas en tiempo real.
- El modelo de datos se hacía de una manera primitiva.

Estas y otras razones dieron nacimiento a ciertas mejoras en el análisis estructurado clásico tales como:

- Diagramas de entidad-relación.
- Diagramas de transición de estados.
- División de eventos.
- Modelos esenciales y
- Modelos de implantación.

Pero a pesar de esto se siguieron dando más problemas como los siguientes:

- Tras la segunda y tercera correcciones de un diagrama, el analista se volvía cada vez más apuesto y renuente a hacer más cambios.
- Debido a la cantidad de trabajo requerido, el analista dejaba a veces de dividir el modelo del sistema en modelos de menor nivel, quedando por ende, funciones primitivas.
- A menudo no se incorporaban en el modelo del sistema los cambios en los requerimientos del usuario sino hasta después de la fase de análisis del proyecto.

Inclusive las correcciones de los diagramas había que hacerlas en forma manual, para asegurar que fuesen consistentes y estuviesen completas; lo cual era bastante tedioso y dejaba por fuera muchos errores que debían de encontrarse. Pero para mediados de la década de 1980 aparecieron las herramientas CASE (Ingeniería de software auxiliada por computadora) que trataron de subsanar estos problemas. Las herramientas CASE se utilizan para dibujar diagramas de flujo de datos, además de llevar acabo una variedad de labores de revisión de errores.

Finalmente, algunos usuarios tenían dificultades al tratar con los modelos gráficos del análisis estructurado y preferían alguna otra forma de modelar los requerimientos y comportamiento del sistema; es por ello que aparecen las herramientas de generación de prototipos (mediados de la década de 1980) que son considerados como una alternativa al análisis estructurado para tales usuarios.

En la actualidad muchas de estas herramientas se están utilizando para facilitar la fase de análisis, e inclusive se están elaborando o fusionando lo mejor de cada una de las técnicas que atienden las necesidades de todas las fases del ciclo de vida del sistema; para así

obtener un mejor aprovechamiento, entendimiento, y rendimiento al momento que se ponga a correr el sistema. Disminuyendo de esta manera la serie de errores que se cometían anteriormente, con la introducción de herramientas más especializadas y fáciles de utilizar.

3.7 Clasificación de las metodologías de desarrollo de software

ENFOQUE	TIPO DE SISTEMA	FORMALIDAD
ESTRUCTURADAS Orientadas a Procesos Orientadas a datos Jerárquicos No jerárquicos Mixtas	GESTIÓN	NO FORMAL
ORIENTADAS A OBJETOS	TIEMPO REAL	FORMAL

Clasificación de las metodologías desde el punto de vista del enfoque

ENFOQUE	CARACTERÍSTICAS	METODOLOGÍA / TÉCNICA	AUTOR	
Estructuradas	Orientadas a procesos	Elaboran modelos de sistemas que se basan en el estudio de los procesos y/o sus entradas y salidas.	Diseño estructurado	Yourdon-Constantine
		Análisis Estructurado	De Marco Gane y Sarson	
		Análisis y diseño estructurado moderno	Yourdon	
	Orientadas a datos jerárquicos	Establecen modelos de sistemas que se basan en una organización y un acceso ideales a los datos del sistema, independientemente de cómo se utilicen estos datos para satisfacer las necesidades de información (salidas).	JSD	Jackson
			Warnier-Orr	Warnier
	Orientadas a datos no jerárquicos		Modelización de datos	Mellor y Shaler
Ingeniería de la información			Martin y Finkelstein	
Mixtas	Combinan los enfoques de orientadas a procesos y orientadas a datos.	SSADM Métrica		
Prototipos	Prototipos de viabilidad	Se utilizan para probar la viabilidad de una tecnología específica aplicable a un sistema de información.		
	Prototipos de necesidades	Durante la elaboración de diseños de necesidades, el analista puede "pintar" pantallas o informes de muestra, y solicitar las opiniones del usuario con respecto a su contenido (pero no su formato).	Prototipos de descubrimiento	
	Prototipos de diseño	Se utilizan para simular el diseño del sistema de información final.	Prototipos de comportamiento	Herramientas y lenguajes de cuarta generación.
	Prototipos de implantación	Constituyen una extensión de los prototipos de diseño donde el prototipo evoluciona directamente hacia el sistema de producción.	Prototipos de producción	
Orientadas	Revolucionarias o puras	Rompen con las metodologías estructuradas.	OOD (Object Oriented Design)	Booch
	Sintetistas o evolutivas	Las metodologías estructuradas son la base para el desarrollo orientado a objetos.	OMT (Object Modeling Technique)	Rumbaugh
	RUP	Es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software.	Método unificado UML	Booch y Rumbaugh
Ágiles	En general los procesos ágiles se centran en las personas; en su comunicación directa y sus habilidades en vez de en procesos muy formales.	eXtreme Programming	Kent Beck	
		Crystal Light Methods	Cockburn	

3.8 Tendencias de las metodologías de desarrollo de software

En un futuro no muy lejano se piensa que se darán, si es que ya no se están dando, los siguientes cambios o pautas en el ámbito total en lo que se refiere a las metodologías de desarrollo de software-

Mayor difusión de las metodologías de desarrollo de software, sobre todo en los siguientes grupos: los niveles superiores de administración en organizaciones gubernamentales y de negocios, y profesionales de la computación en los países del tercer mundo.

- Impacto sobre la industria de software del tercer mundo.
- Proliferación de las herramientas automatizadas.
- Impacto de los desastres de mantenimiento.
- Integración de las metodologías de desarrollo de software con la inteligencia artificial.
- Proliferación de metodologías que se basan en otros aspectos, además de los procesos o los datos.
- Mayor auge en las metodologías basadas en las personas “metodologías ágiles de desarrollo”.
- Mayor auge en metodologías que fortalecen la reutilización de software como lo son las “metodologías orientadas a objetos”.

Podemos adicionar que el futuro de las metodologías de desarrollo va a depender también de que tan rápido puedan ajustarse a los cambios tecnológicos que se viven hoy en día.

Al día de hoy, ha aumentado la complejidad con la que se desarrollan sistemas de información para la industria, por lo que resulta difícil generar productos que cumplan cabalmente con las expectativas del cliente.

Para responder a esta situación, han surgido una serie de herramientas, técnicas y modelos que facilitan a las organizaciones, encargadas de las tecnologías de la información, generar productos que cumplan las expectativas del cliente e incluso las rebasen, herramientas que prometen ser la solución a los problemas de calidad, costo y tiempos de desarrollo; de éstas podemos mencionar a los (modelos de calidad) como la norma ISO 9000-2000, la ISO/IEC TR 15504 y el modelo Capability Maturity Model (CMM) del Software Engineering Institute (SEI).

Aunque en el pasado se reconocía la necesidad de crear software de calidad, no se había hecho un esfuerzo serio para que la industria en México generara productos que nos dieran la oportunidad de competir en el mercado internacional, con calidad equiparable o superior a la de países como la India o Irlanda.

Afortunadamente, dicha situación ha cambiado; En 2002 la Secretaría de Economía (SE) inició el Programa para el Desarrollo de la Industria de Software (PROSOFT), que tiene como objetivo Fortalecer a la Industria de Software²⁵.

PROSOFT reconoce el estado incipiente de la industria mexicana de software, así como la necesidad de invertir cantidades crecientes de recursos en capital de tecnologías de información con objeto de contribuir de manera sostenible al crecimiento de la economía y la generación de empleos bien remunerados.

²⁵ Liga de PROSOFT: www.software.net.mx

4 MARCO METODOLÓGICO

4.1 Variables

La investigación realizada en el marco problemático y el marco conceptual del presente trabajo nos permitió llegar a la conclusión de que las variables en su relación causa-efecto subsisten de la siguiente manera.

Variable independiente:

- Carencia de conocimientos sobre los aspectos importantes al elegir una metodología de desarrollo de software que cubra las necesidades específicas del proyecto a desarrollar.

Variables dependientes:

- Un desarrollo de software impredecible e ineficiente, los tiempos y resultados del desarrollo del sistema no coinciden con lo estipulado.
- Inconformidad e insatisfacción del cliente.

4.2 Variables de control

Debido al trabajo realizado en el marco problemático y el marco conceptual llegamos a la conclusión de que no se incluyen variables de control, de tipo interviniente, ni distorsionante, puesto que no se identificaron algunos otros factores que puedan ser relevantes para esta investigación.

4.3 Definición del universo

El universo considerado para esta investigación que tiene como propósito el de aprobar una hipótesis, a través de la obtención de conocimientos y la concordancia que exista de nuestro criterio hipotético, con el de las opiniones calificadas y experimentadas en el uso de las metodologías de desarrollo de software, esta integrado por empresas e instituciones que se dedican a desarrollar sistemas utilizando alguna metodología de desarrollo de software.

4.4 Determinación de la muestra

Se utilizó una muestra no probabilística, denominada de juicio o intencionada, ya que como se mencionó anteriormente nuestro estudio no busca hacer una prueba plena de la hipótesis, sino obtener una relación válida, por lo que se entrevistó a 15 personas calificadas en el

conocimiento de metodologías de desarrollo de software e involucradas con el desarrollo de software, tanto de organizaciones publicas como privadas, de diferentes roles, aceptando su opinión como valida.

4.5 Definición del método de la investigación

El instrumentos de investigación que utilizamos es el cuestionario, el cuál se aplicó personalmente y en algunos casos vía correo electrónico. Se utilizó éste instrumento por la facilidad y rapidez con que se obtiene la información y teniendo en cuenta que el perfil de las personas que lo contestaron les permite hacerlo de forma eficaz. El cuestionario esta integrado por 12 preguntas, la mayoría cerradas, para facilitar y precisar las respuestas de los encuestados. No obstante se permitió completar con comentarios adicionales.

4.6 Costo de la investigación

CONCEPTO	COSTO
Recursos Humanos (2 personas en 9 meses con un sueldo mensual de \$ 8,000.00).	\$144, 000.00
Renta (Lugar de trabajo).	\$ 18, 000.00
Material de investigación (libros, revistas, fotocopiado).	\$ 1, 500.00
Uso de equipo de cómputo.	\$ 5, 000.00
Papelería y artículos diversos de computo	\$ 1, 200.00
Comunicación (teléfono, Internet).	\$ 2, 000.00
Impresión y terminado.	\$ 5, 000.00
Otros varios (Transportación a los sitios de recopilación de información, etc.).	\$ 3, 300.00
Total	\$180, 000.00

4.7 Cuestionario

El objetivo de este cuestionario es comprobar si el uso de una metodología de desarrollo de software es indispensable, para el buen desarrollo de sistemas informáticos. Ver anexo E.

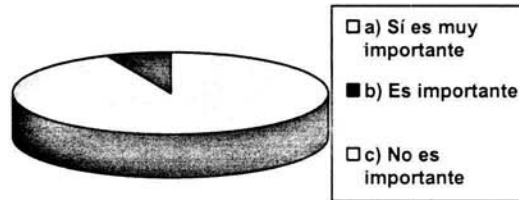
4.8 Personas entrevistadas

Nombre	Iniciales	Puesto	Organización
Jacobo Alexander Cano González	JACG	Analista-programador	Instituto de Vivienda del Distrito Federal
L. I. Gabriel Guevara Gutiérrez	GGG	Analista de sistemas	Centro de Informática de la Facultad de Contaduría y Administración (CIFCA) UNAM.
Alfonso Jiménez Lara	AJL	Consultor	Structured Intelligence.
Sonia Carolina Madrigal Loyola	SCML	Analista-programador	Secretaría de Medio Ambiente y Recursos Naturales y en la Organización de las Naciones Unidas
L. I. Cesar Augusto Morones Sánchez	CAMS	Desarrollador multimedia	EDUWEB
Enrique Phillips Márquez	EPM	Administrador de proyectos	Optimización Administrativa, S. A. de C. V.
Ing. Angélica María Ramírez Bedolla	AMRB	Analista programador	Laboratorio de multimedia de la DGSCA-UNAM.
Felipe Ramírez de la O.	FRO	Líder de proyecto	Elektra (Banco Azteca).
L. I Luz María Ramírez Romero	LMRR	Analista de sistemas	Coordinación de Servicios de red de la DGSCA-UNAM.
MTIA. Hugo Alonso Reyes Herrera	HARH	Administrador de proyectos	Subdirección de Sistemas de la Dirección de Sistemas de DGSCA-UNAM.
Ing. Edward Rodríguez Zavala.	ERZ	Ingeniero 3	CERTUM. Servicios en informática y desarrollo de México, S.A. de C. V.
Francisco Serna Torres	FST	Administrador de proyectos	Software Architect. Smart it Solutions.
Act. Alejandro Talavera Rosales	ATR	Web Master	Laboratorio de multimedia de la DGSCA-UNAM.
L. I. Nora Elizabeth Tapia Ruíz	NETR	Ingeniero de pruebas	Subdirección de Sistemas de la Dirección de Sistemas de DGSCA-UNAM
L. I. María Teresa Ventura Miranda	MTVM	Analista de sistemas	Subdirección de Sistemas de la Dirección de Sistemas de DGSCA-UNAM

4.9 Análisis de la información recopilada

PREGUNTA: 1.- ¿Considera que las metodologías para el desarrollo de software son un factor primordial para el éxito de un proyecto?

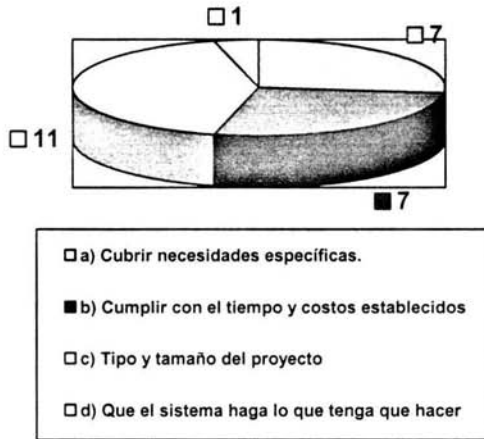
Presentamos la recopilación y el análisis de la información del cuestionario final en el cual el entrevistado tuvo la oportunidad de elegir una o más opciones en cada respuesta.



De 14 entrevistados 15 consideran que las metodologías para el desarrollo son *muy importantes* como factor primordial para el éxito de un proyecto. Sólo uno opina que las metodologías para el desarrollo son *importantes* como factor primordial para el éxito de un proyecto.

Los resultados obtenidos por esta pregunta respaldan en un 93% la temática de nuestra investigación, al establecer que la mayoría de los entrevistados concuerdan que las metodologías para el desarrollo de software son muy importantes, así podemos concluir que las metodologías para desarrollo de software son un factor primordial para el éxito de un proyecto.

2.- ¿Cuál considera que sean los aspectos más importantes para la elección de una metodología para el desarrollo de software?

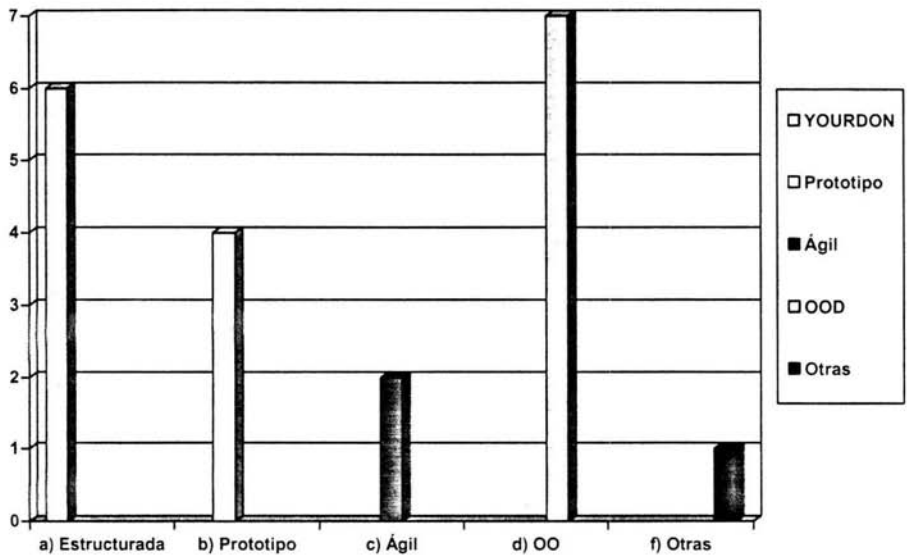


Los entrevistados consideran que los aspectos más importantes para la elección de una metodología para el desarrollo de software son:

- La mayoría considera que el tipo y tamaño del proyecto es el más importante.
- Y en segundo lugar opinan que cubrir necesidades específicas y cumplir con el tiempo y costos establecidos.

Con esta información concluimos que nuestra hipótesis se comprueba al 73.3 %, ya que 11 de 15 personas coincidieron en elegir entre sus respuestas el tipo y tamaño del proyecto, cubrir necesidades específicas de cada proyecto y cumplir con el tiempo y costos establecidos como aspectos muy importantes para la elección de las metodologías para el desarrollo de software, que permiten generar sistemas predecibles y eficientes.

3.- ¿Utiliza alguna metodología para desarrollo de software?

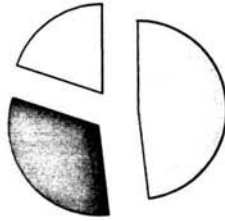


Los entrevistados utilizan las siguientes metodologías para el desarrollo de software, cabe aclarar que algunos utilizan más de una metodología; pues depende del proyecto a desarrollar.

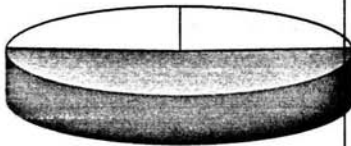
En esta matriz identificamos las metodologías de desarrollo de software que son más utilizadas en la actualidad en nuestro universo definido. En este caso la mayoría de los entrevistados utilizan la metodología orientada a objetos p. ej.: RUP para el desarrollo de software, así como la metodología estructurada. Solo una persona dice que utiliza la metodología Team Software Process (TSP).

Por lo que se puede ver existe gran diversidad de metodologías que se utilizan, de esta forma podremos deducir de posteriores respuestas cuáles son las razones que determinan el uso de una metodología para desarrollo de software.

4.- ¿Qué tipo de proyectos desarrolla con esta metodología para el desarrollo de software?



- a) Sistemas complejos
- b) Sistemas medianos
- c) Sistemas pequeños



- a) Crítico
- b) Administrativo
- c) Comercial

En éste cuadro podemos observar que nuestros encuestados desarrollan mayoritariamente sistemas complejos y de tipo administrativo. A continuación presentamos una matriz en donde clasificamos esta información por tipo de metodología para un estudio más completo.

Matriz metodología / tamaño y tipo de sistema							
Metodologías/personas Personas que eligieron cada metodología	Tamaño				Tipo		
	Sistemas complejos	Sistemas medianos	Sistemas pequeños	Crítico	Administrativo	Comercial	
Estructurada	6	4	4	2		6	1
Prototipo	4	3	1	1	1	2	3
Ágil	2	2	1		1	1	
OO	7	6	5	3	3	7	2
Otras:	1	1			1		

Podemos deducir que:

- La metodología estructurada se utiliza en un 66% para desarrollar sistemas complejos y sistemas medianos y 100% para tipo administrativo con respecto al número de personas que la utilizan.

- La metodología prototipo se utiliza en un 75% para desarrollar sistemas complejos, de la misma forma el 75% para tipo comercial con respeto al número de personas que la utilizan.
- La metodología orientada a objetos se utiliza para desarrollar sistemas complejos y medianos, de la misma forma en un 100% para sistemas tipo administrativo con respeto al número de personas que la utilizan.
- La metodología ágil se utiliza para desarrollar sistemas complejos y medianos, así como de tipo administrativo y comercial con respeto al número de personas que la utilizan.
- La metodología TSP (Team Software Process) se utiliza para desarrollar sistemas complejos y de tipo crítico.

De lo anterior podemos concluir que efectivamente el tipo y tamaño del proyecto puede determinar el uso de una metodología para el desarrollo de software en particular, cabe destacar el amplio desenvolvimiento que tienen la metodologías orientadas a objetos y la estructurada en proyectos complejos y de tipo administrativo, así como la de prototipo para el mismo tamaño de proyectos pero difiere en que se usa para proyectos de tipo comercial.

5.- ¿Por qué utiliza esta metodología de desarrollo de software?					
	a) Estructurada (Yourdon)	b) Prototipo	c) Ágil (XP)	d) OO (RUP)	e) Otras
JACG	Ha funcionado bien en desarrollos anteriores				
GGG	Ha dado buenos resultados en productos finales.	Ha dado buenos resultados en productos finales.		Ha dado buenos resultados en productos finales.	
AJL		Nos permiten reducir riesgos, costos, incrementar nuestra productividad, y liberar aplicaciones 100% confiables.		Nos permiten reducir riesgos, costos, incrementar nuestra productividad, y liberar aplicaciones 100% confiables.	
SCML	Satisface las necesidades de los proyectos.				
CAMS		Porque los sistemas son pequeños y por la falta de seriedad del			

		cliente.			
EPM	Porque asegura la terminación exitosa del proyecto.				
AMRB		El desarrollo de prototipos permite mostrar al cliente su idea plasmada en una aplicación de software.			
FRO				Para organizar mejor el trabajo, evitar trabajos dobles, calidad en la documentación.	
LMRR	Ha dado buenos resultados en productos finales.		Ha dado buenos resultados en productos finales.		
HARH				Permite una mejor planificación de tiempos y costos y claridad en el alcance. Permite cumplir con mayor calidad el objetivo del proyecto.	
ERZ			Porque es la que mejor se adapta al tipo desarrollo que se llevan a acabo en esta empresa.		
FST				Estimula la reutilización de componentes con lo cual se logra mayor eficiencia.	
ATR					Por la división del trabajo. Es cíclico y evolutivo.
NETR				Por ser	

				descriptiva, orientada a las operaciones y procesos. Ser un estándar a nivel mundial.	
MTVM	Ha sido probada y aceptada dentro de la organización. Promueve la participación del cliente.			Ha sido probada y aceptada dentro de la organización. Promueve la participación del cliente.	

Con esta información podemos confirmar la importancia de tomar en cuenta los aspectos importantes como tipo y tamaño de proyecto entre otras, al elegir una metodología para desarrollar proyectos específicos, con lo cual se aprueba nuestra hipótesis en un 100% ya que esto permite obtener buenos resultados.

6.- ¿Desde cuándo utiliza esta metodología de desarrollo de software?

	a) Estructurada (Yourdon)	b) Prototipo	c) Ágil (XP)	d) OO (RUP)	e) Otras
JACG	4 años				
GGG	4 años	4 años		3 años	
AJL		7 años		7 años	
SCML	5 años				
CAMS		1 año			
EPM	Desde 1970				
AMRB		4 años			
FRO				6 meses	
LMRR	10 años		1 año		
HARH				4 años	
ERZ			10 años		
FST				6 años	
ATR					2 años
NETR				4 años	
MTVM	4 años			4 años	

Con los resultados de esta pregunta podemos confirmar el buen funcionamiento de una metodología ya que con los años que ha sido utilizada esa metodología se puede ver si es ya parte de cómo hacer las cosas en esa organización. Podemos decir que la metodología estructurada se ha venido utilizando desde hace más tiempo, siendo la base para forjar nuevas metodologías; que prototipo y orientada a objetos se están estableciendo para ser utilizadas actualmente.

7.- ¿Aplica formalmente todos los elementos de la notación de esta metodología?					
	a) Estructurada (Yourdon)	b) Prototipo	c) Ágil (XP)	d) OO (RUP)	e) Otras
JACG	No, porque se pueden omitir algunos elementos sin afectar el desempeño.				
GGG	No, dado a que son modelos extranjeros.	No, dado a que son modelos extranjeros.		No, dado a que son modelos extranjeros.	
AJL		Sí, la mayoría de las veces.		Sí, la mayoría de las veces.	
SCML	No, a veces el tiempo no lo permite.				
CAMS		Sí			
EPM	No sé si conozco todos los pasos de esa metodología.				
AMRB		Sí, aunque con limitaciones, ya que en la fase de pruebas no continua después de que el producto es lanzado comercialmente.			
FRO				No, por los tiempos cortos y la dinámica del negocio.	
LMRR	No, no definimos todos los flujos de datos entre procesos en el diccionario de datos, tampoco hacemos mini especificaciones		No, no definimos todos los flujos de datos entre procesos en el diccionario de datos, tampoco hacemos mini especificaciones		
HARH				No, creo que la metodología se aplica de acuerdo a las necesidades de cada entidad.	

ERZ			No, porque aquí uno de los aspectos más importantes es el tiempo.		
FST				Sí, usamos una adaptación de una metodología orientada a objetos y usando UML como base.	
ATR					Sí, se cumple con los formularios y métricas.
NETR				No, de acuerdo al tiempo que se tenga para cada proyecto y la complejidad de cada uno.	
MTVM	Sí, se procura respetar la notación.			Sí, se procura respetar la notación.	

La mayoría de los entrevistados afirman el hecho de que no aplican formalmente todos los elementos de la notación de la metodología que utilizan debido a diversas razones como son: tiempo, costo, conocimientos de las personas y porque son modelos extranjeros no aplican en su totalidad, además se pueden omitir algunos elementos sin afectar el desempeño.

8. -¿Específicamente para qué usa esta metodología para el desarrollo de software?					
Respuestas:	a) Planificación y control del proyecto. b) Mejores aplicaciones. c) Proceso estándar para una comunicación efectiva. d) Para tener la documentación del proyecto. e) No tengo una clara idea de porque se utiliza esta metodología. f) Otras.				
Metodologías	a) Estructurada	b) Prototipo	c) Ágil (XP)	d) OO (RUP)	e) Otras
JACG	d.				
GGG	a, c, d.	a, c, d.		a, c, d.	
AJL	a, b, c, f: Administración de riesgos, control de cambios.	a, b, c, f: Administración de riesgos, control de cambios.		a, b, c, f: Administración de riesgos, control de cambios.	
SCML	a, b, c, d.				
CAMS		e.			
EPM	a, f: Permite asegurar que los entregables cumplan con lo comprometido con el cliente.				
AMRB		c.			
FRO				a, c, d, f: Para que la empresa pueda certificarse.	
LMRR	c, d.		c, d.		
HARH				a, c, d.	
ERZ			a.		
FST				a, b, c, d.	
ATR					a.
NETR				a, c, d.	
MTVM	f: Para especificar y validar requerimientos promoviendo el involucramiento del cliente.			f: Para especificar y validar requerimientos promoviendo el involucramiento del cliente.	

Casi todos los entrevistados coinciden en que la razón por la que utilizan una metodología para el desarrollo de software es: la planificación y el control del proyecto. En el caso más específico los entrevistados que usan las metodologías orientadas a objetos y la estructurada dicen que sus razones son el proceso estándar para una comunicación efectiva, promoviendo el involucramiento del cliente y para tener la documentación del proyecto.

Con esta información se desprende la aprobación de nuestra hipótesis ya que los entrevistados identifican los beneficios más importantes por los que se utilizan las metodologías para el desarrollo de software, estos beneficios permiten que se obtenga nuestra variable dependiente como son: la planificación y el control del proyecto, el proceso estándar para una comunicación efectiva, el involucramiento del cliente y para tener la documentación del proyecto.

9.- ¿Cuáles son las ventajas de la metodología que usted utiliza?					
Respuestas:	a) Seguimiento ordenado y completo. b) Existencia de reglas predefinidas. c) Cobertura total del ciclo de vida. d) Verificaciones intermedias. e) Planificación y control del proyecto. f) Comunicación efectiva. g) Utilización sobre un abanico amplio de proyectos. h) Fácil formación. i) Soporte de la reutilización del software. j) Actividades que mejoren el proceso. k) Soporte al mantenimiento. l) Soporte de la reutilización de software. m) No hay ventajas. n) Otras:				
Metodologías	a) Estructurada	b) Prototipo	c) Ágil (XP)	d) OO (RUP)	e) Otras
JACG	a, f.				
GGG	a, b, c, e, f, k, l.	a, b, c, e, f, k, l.		a, b, c, e, f, k, l.	
AJL	a, b, c, d, f, g, h, j, k.	a, b, c, d, f, g, h, j, k.		a, b, c, d, f, g, h, j, k.	
SCML	a, b, d, f, k.				
CAMS		m.			
EPM	a, c, d, e, f, g.				
AMRB		d, f, l.			
FRO				a, c, e, f, i, k, n: Ciclo iterativo de desarrollo por fases.	
LMRR	d, f, j, k, n: Ayudan a satisfacer de mejor manera las necesidades del usuario.		d, f, j, k, n: Ayudan a satisfacer de mejor manera las necesidades del usuario.		
HARH				a, c, e, f, l.	
ERZ			d.		
FST				c, e, f, g, i, j, k, l.	
ATR					a, c, e, j, k, l.

NETR				b, d, e, f, j, n: Mejorar el proceso de desarrollo y el de cada proyecto.	
MTVM	d, e, f, i, n: Negociación con el cliente y los usuarios.			d, e, f, i, n: Negociación con el cliente y los usuarios.	

Casi todos los entrevistados enumeran la diversas ventajas que les otorga el uso de una metodología de desarrollo de software destacando entre ellas: Seguimiento ordenado y completo, verificaciones intermedias, comunicación efectiva, soporte al mantenimiento; además integran otra muy importante como lo es la comunicación cercana y el involucramiento con el cliente.

10.- ¿Cuáles son las desventajas de usar la metodología que usted utiliza?

Respuestas:	a) Dificultad para especificar claramente los requerimientos. b) No permite flexibilidad en los cambios. c) Requiere conocimientos y experiencia altamente especializados. d) No proporciona resultados tangibles en forma de software. e) Los usuarios de las metodologías los interpretan según su punto de vista. f) Saber cuándo dejar de refinar y cuándo consentir con el sistema. g) Imposibilidad de conocer al comienzo del proyecto el tiempo y costo. h) No hay desventajas. i) Otras.				
Metodologías	a) Estructurada	b) Prototipo	c) Ágil (XP)	d) OO (RUP)	e) Otras
JACG	e.				
GGG				c.	
AJL	h.	h.		h.	
SCML	e.				
CAMS		i: Es difícil determinar las características y metodologías apropiadas.			
EPM	e.				
AMRB		d, i: Es difícil conocer tiempos y costos, a menos que estos sean especificados por el cliente.			
FRO				c, i: Altos	

				costos de las herramientas CASE y su capacitación en ellas.	
LMRR	b.		g.		
HARH				c.	
ERZ			i: A veces por la premura de tiempo, se tenga que utilizar más recurso humano que el estimado.		
FST				c.	
ATR					d, f, i: Puede resultar ser muy burocrático al principio.
NETR				c, i: Al momento de utilizar toda la metodología es compleja para terminar toda la documentación	
MTVM	i: Se especifica una parte en "lenguaje natural", lo que puede prestarse a ambigüedades.			i: Se especifica una parte en "lenguaje natural", lo que puede prestarse a ambigüedades.	

Las desventajas que enumeran los entrevistados son diversas de ello se puede deducir que todos los proyectos tienen necesidades fundamentales que se deben cumplir y que a veces una metodología de desarrollo de software no puede hacerlo debido a que es un esquema rígido.

Se concluye además que se tenga que recurrir al uso de diversas metodologías en conjunto para intentar cubrir todas sus necesidades; incluso se puede tratar de implementar su propia forma de hacer las cosas.

11.- ¿Ha sido complicado incorporar esta metodología?					
	a) Estructurada (Yourdon)	b) Prototipo	c) Ágil (XP)	d) OO (RUP)	e) Otras
JACG	No, debido a la utilización de la metodología anteriormente.				
GGG	Sí, los servidores sociales tienen poco o nulo conocimiento.	Sí, los servidores sociales tienen poco o nulo conocimiento.		Sí, los servidores sociales tienen poco o nulo conocimiento.	
AJL		No, con la capacitación adecuada.		No, con la capacitación adecuada.	
SCML	No, es cuestión de que los participantes del proyecto vean las ventajas de seguir una metodología.				
CAMS		Sí, porque no se planea nada.			
EPM	No.				
AMRB		No, ha sido bueno utilizar una metodología que se adopte al desarrollo de proyectos de multimedia.			
FRO				Sí, todo cambio en la forma de trabajo de cualquiera es complicado implantarlo.	
LMRR			No, los desarrolladores han aceptado bastante bien XP.		
HARH				No, en un principio romper el paradigma de lo estructurado fue muy complejo para quienes ya	

				teníamos varios años en la programación. Para los que van empezando, la transición fue casi transparente.	
ERZ			No, Porque se ha ido incorporando conforme crece la empresa.		
FST				No, cuando se creo la empresa, casi todo el personal ya contaba con los conceptos básicos.	
ATR					Si, se requiere que el equipo completo tenga el mismo nivel de compromiso.
NETR				Si, ha sido un proceso evolutivo y de mejora, así como enseñar a cada persona su interpretación y aplicación.	
MTVM	No, es sencilla en la parte de análisis.			No, es sencilla en la parte de análisis.	

La mayoría de los entrevistados nos demuestran que el factor determinante para incorporar una metodología de desarrollo de software es la gente, es decir, las personas que van a utilizarla por ello se deduce que a medida que se tenga el personal adecuado (personal con conocimientos y experiencia), la metodología de desarrollo de software va a ser fácilmente incorporada y entendida por todos en la organización, y así se podrá optimizar su uso.

12.- ¿Cree usted que existe una metodología para el desarrollo de software eficaz para cada tipo de proyectos?					
	a) Estructurada (Yourdon)	b) Prototipo	c) Agil (XP)	d) OO (RUP)	e) Otras
JACG	Sí.				
GGG	Sí.	Sí.		Sí.	
AJL		Sí.		Sí.	
SCML	Sí.				
CAMS		Sí.			
EPM	No, lo más importante es tener una metodología y seguirla siempre, sólo así se asegura el resultado final, a veces en mayor tiempo o con mayor esfuerzo.				
AMRB		No, definitivamente cada desarrollo debe tomar una o varias partes de una o varias metodologías de acuerdo a las necesidades de los proyectos.			
FRO				Sí, para todo proyecto habrá una metodología mas a la medida que cumpla con el objetivo de que el proyecto se termine lo mejor posible.	
LMRR			Sí.		
HARH				Si, yo creo que por ejemplo RUP incorpora algunas de las mejores prácticas para una gran variedad de proyectos, pero no podríamos	

				hablar que una herramienta o metodología es aplicable a cualquier tipo de proyecto.	
ERZ			No sé.		
FST				No, lo importante es tomar una, y adaptarla a cada tipo de proyecto.	
ATR					Si.
NETR				No, creó que la experiencia es importante y las metodologías son una guía, pero no aseguran el éxito del proyecto, debe existir una sinergia de metodología, personal capacitado, inversión en T. I., etc.	
MTVM	No, no hay una metodología universal. Hay muchas que son muy buenas y han sido ampliamente probadas. No obstante lo más efectivo es retomar lo mejor de éstas o de una en particular y aplicarla al contexto de la organización del tipo del proyecto, etc. Es decir adaptarla a la situación particular.			No, no hay una metodología universal. Hay muchas que son muy buenas y han sido ampliamente probadas. No obstante lo más efectivo es retomar lo mejor de éstas o de una en particular y aplicarla al contexto de la organización del tipo del proyecto, etc. Es decir adaptarla a la situación particular.	

8 de 15 personas nos contestan que si existe una metodología para el desarrollo de software eficaz para cada tipo de proyectos, las otras 7 personas mencionan diferentes variables al utilizar una metodología específica como son:

- Tener una metodología y seguirla siempre, sólo así se asegura el resultado final, a veces en mayor tiempo o con mayor esfuerzo.
- No hay una metodología universal. Hay muchas metodologías que son muy buenas y han sido ampliamente probadas, no obstante lo más efectivo es retomar lo mejor de éstas o de una en particular y aplicarla al contexto de la organización y del tipo del proyecto.
- Las metodologías son una guía, pero no aseguran el éxito del proyecto, debe existir una sinergia de metodología, personal capacitado, inversión en T. I.

4.10 Conclusión sobre los resultados

Podemos afirmar que existe conciencia de lo importante que es emplear una metodología de desarrollo de software adecuada al tipo de proyecto a realizar, ya que esto implica crear productos de calidad optimizando recursos para satisfacer las necesidades de los clientes y usuarios.

La importancia de las metodologías de desarrollo de software radica en que permite la planificación y el control del proyecto, permitiendo con esto un seguimiento ordenado y completo, verificaciones intermedias, comunicación efectiva, soporte al mantenimiento; además permite la comunicación cercana e involucramiento con el cliente.

A pesar de que los encuestados utilizan alguna metodología de desarrollo de software, tienen diferentes problemáticas al desarrollar sistemas de información dependiendo de la metodología que utilicen, la mayoría de los entrevistados afirman el hecho de que no aplican formalmente todos los elementos de la notación de la metodología debido a diversas razones como son: tiempo, costo, conocimientos de las personas y porque son modelos extranjeros que no aplican en su totalidad, además se pueden omitir algunos elementos sin afectar el desempeño. Las desventajas que enumeran los entrevistados son diversas de ello se puede deducir que todos los proyectos tienen necesidades fundamentales que se deben cumplir y que a veces una metodología de desarrollo de software no puede hacerlo debido a que es un esquema rígido.

Algunos entrevistados opinan que no hay una metodología universal; hay muchas que son muy buenas y han sido ampliamente probadas no obstante lo más efectivo es retomar lo mejor de éstas o de una en particular y aplicarla al contexto del tipo de proyecto y de la organización. Es decir adaptarla a la situación particular, incluso pueden ser utilizadas varias metodologías para planear, desarrollar, concluir y/o mantener el sistema. Con esto concluimos que se tenga que recurrir al uso de diversas metodologías en conjunto para intentar cubrir todas sus necesidades; incluso se puede tratar de implementar su propia forma de hacer las cosas.

Un factor determinante para incorporar una metodología de desarrollo de software es la gente, es decir, las personas que van a utilizarla por ello se deduce que a medida que se tenga el personal adecuado (personal con conocimientos y experiencia), la metodología de desarrollo de software va a ser fácilmente incorporada y entendida por todos en la organización, y así se podrá optimizar su uso.

Concluimos que es de vital importancia tener conocimientos sobre las metodologías de desarrollo de software para elegir la óptima para cada proyecto, basándose en aspectos relevantes como son: tipo y tamaño de proyecto, necesidades específicas; esto nos permitirá crear un producto de calidad que satisfaga al cliente y/o usuario.

Con esta información se desprende la aprobación de nuestra hipótesis ya que los entrevistados identifican los beneficios más importantes por los que se utilizan las metodologías para el desarrollo de software (estos beneficios permiten que se obtenga nuestra variable dependiente) como son: la planificación y el control del proyecto, el proceso estándar para una comunicación efectiva, el involucramiento del cliente y para tener la documentación del proyecto.

4.11 Aprobación de la hipótesis

Para nosotras es de suma importancia llegar a la conclusión de que actualmente en las empresas u organizaciones, las metodologías de desarrollo de software tienen gran aceptación, y que además, se reconoce que son de vital importancia, ya que funcionan como un elemento clave para llegar a un producto de calidad que pueda cumplir con las expectativas deseadas.

En las organizaciones las personas encargadas de los proyectos informáticos reconocen el valor que tiene las metodologías de desarrollo de software al elegir una adecuada y de los aspectos en que se tienen que basar para dicha elección, teniendo también en cuenta que los elementos de esa metodología no serán llevados al 100 %, más bien se tratará de adaptar a sus necesidades específicas, siempre con un sólo objetivo; la satisfacción del cliente.

Con el trabajo realizado a lo largo de este capítulo, después de haber delimitado el problema y ubicado nuestro universo, se aplicaron varios cuestionarios con los que obtuvimos muchas opiniones acertadas acerca de las metodologías de desarrollo de software.

Con base en el análisis de los resultados y las conclusiones obtenidas de ello, podemos decir que nuestra *hipótesis es aprobada* por las razones que explicaremos a continuación:

La hipótesis:

“La carencia de conocimientos sobre aspectos importantes al elegir una metodología de desarrollo de software, que cubra las necesidades específicas del proyecto a desarrollar, origina un desarrollo de software carente de grado de predicibilidad y eficiencia, los tiempos y resultados del desarrollo no coinciden con lo estipulado, generando así inconformidad e insatisfacción del cliente”.

Los resultados:

Los resultados obtenidos por la pregunta 1 del cuestionario final respaldan en un 93% nuestra hipótesis, al establecer que la mayoría de los entrevistados coinciden en que las metodologías para el desarrollo de software son *muy importantes*, así podemos concluir que

las metodologías para desarrollo de software son un factor primordial para el éxito de un proyecto.

Con base en la respuestas de la pregunta 3 del cuestionario final se comprueba nuestra hipótesis en un 100 % ya que todos nuestros entrevistados utilizan por lo menos alguna metodología de desarrollo de software, en algunos casos más de una.

Las personas entrevistadas coinciden con nuestro planteamiento de aspectos importantes al elegir una metodología para el desarrollo de software en la siguiente proporción:

- El 73.3. % afirma que el tipo y tamaño de proyecto es uno de los aspectos más importantes.
- En segundo lugar identifican cubrir necesidades específicas y el cumplir con el tiempo y costos establecidos como los aspectos más importantes.

La mayoría de las personas entrevistadas conocen acerca de los aspectos importantes que tienen que considerar al elegir una metodología para el desarrollo de software, porque el único objetivo que buscan en realidad es producir software de calidad y que cumpla con las expectativas del cliente.

5 MARCO INSTRUMENTAL

5.1 Propuestas de acción

Debido a la importancia que tiene la temática de nuestro trabajo de investigación sobre *Metodologías de Desarrollo de Software* en el área de informática, creemos que el difundir información sobre el tema será de gran utilidad para la comunidad informática para lograr este objetivo, se exponen a continuación las tareas a realizar:

➤ **Página en Internet**

Con toda la información recabada a lo largo de nuestra investigación nos dimos a la tarea de elaborar una página en Internet que cubriera con el objetivo de presentar información actualizada y de interés para la comunidad informática, ya que con el trabajo de investigación realizado nos dimos cuenta que las metodologías para desarrollo de software son un factor primordial en la creación de sistemas o proyectos informáticos dentro de las organizaciones.

De manera general nuestro sitio de Internet contiene información útil referente a nuestro tema de investigación: metodologías para desarrollo de software.

➤ **Publicación de artículo**

Se presentó una propuesta para la publicación de un artículo, con el propósito de dar a conocer de manera general a la comunidad de la Facultad de Contaduría y Administración de la Universidad Nacional Autónoma de México y en especial a los estudiantes que cursan la Licenciatura en Informática, lo que son las metodologías de desarrollo de software y para que sirven, la importancia que tiene el conocimiento sobre los aspectos que se deben tomar en cuenta al elegir metodologías.

➤ **Inclusión temática en el programa de estudios**

Se elaboró una la propuesta de contenido para la creación de una materia optativa sobre *Metodologías de Desarrollo de Software*, que se presentó a la coordinación de informática de la Facultad de Contaduría y Administración de la Universidad Nacional Autónoma de México, para complementar el plan de estudio de la Licenciatura en Informática, con el firme propósito de que esta temática sirva como una herramienta que los egresados obtengan durante su preparación universitaria y que pueda cubrir esta importante área de su desarrollo profesional.

➤ **Presentación ante grupos**

Con todos los conocimientos que adquirimos durante nuestro trabajo de investigación sobre *Metodologías de Desarrollo de Software*, estamos en la mejor disposición para participar en alguno de los eventos que se llevan a cabo en la Facultad de Contaduría y Administración de la Universidad Nacional Autónoma de México, tales como la semana de cómputo, feria del libro, emprendedores, entre otros; ofreciendo cursos y conferencias sobre nuestro tema de investigación complementándolo con la experiencia profesional que se adquiera en esta área.

5.2 Plan y programa de trabajo

➤ **Página en Internet**

El sitio se encuentra en: http://es.geocities.com/met_des_soft/ y cuenta con los siguientes módulos. Ver anexo F.

Home: Es una bienvenida al sitio y una introducción a la temática que se aborda.

Historia: Presenta una breve evolución cronológica del tratamiento mecánico y electromecánico, aunado al origen y evolución de las metodologías de desarrollo de software.

Evolución: Trata la evolución de las metodologías de desarrollo de software.

Clasificación: Presenta una clasificación de diversas metodologías por enfoque de desarrollo de sistemas.

Ligas de interés: En ésta sección se encuentran ligas referentes a las metodologías de desarrollo de software.

Quiénes somos: En este apartado se encuentra nuestra información con el fin de que puedan contactarnos.

➤ **Publicación de artículo**

Respecto a la publicación de un artículo, este fue entregado al Secretario de Divulgación y Fomento Editorial de la Facultad de Contaduría y Administración de la Universidad Nacional Autónoma de México, el día 24 de Junio de 2004, teniendo como respuesta que el artículo se sometería a revisión y que de acuerdo sus características se planearía su publicación en alguno de los próximos números de agosto de la publicación "Academia". Ver carta de presentación en anexo G.

El artículo tiene como objetivo principal el de despertar el interés por el tema de las *Metodologías de Desarrollo de Software* a los estudiantes de Informática, de ésta forma fortalecer su desempeño en el área laboral al permitir el que se sepa retomar lo mejor de

cada una de estas metodologías o de una en particular, para adaptar a la situación particular que vive cada organización. Ver artículo en anexo H.

➤ **Inclusión temática en el programa de estudios**

El día 28 de Junio de 2004, se entregó en la jefatura de la División de Informática, la propuesta de contenido para la creación de una materia optativa sobre *Metodologías de Desarrollo de Software*, considerando que es un elemento primordial en la formación del licenciado en informática. Ver carta de presentación en anexo I.

La propuesta de contenido para la creación de una materia optativa sobre *Metodologías de Desarrollo de Software* tiene como objetivo principal el que los alumnos aprenderán a utilizar las metodologías, técnicas y estándares necesarios para definir la especificación de un sistema y participar en la gestión del desarrollo y el aseguramiento de la calidad de un producto software. Ver temario para materia optativa en anexo J.

CONCLUSIONES

Conclusiones del marco contextual

En el Marco contextual:

- 1) Existe una problemática en las organizaciones donde es realizado el desarrollo de sistemas de software, ya que generalmente no se entrega a tiempo el producto, esto repercute en el costo del mismo y por ello son afectadas las expectativas del cliente.
- 2) Llegamos a la conclusión de que el personal dedicado al desarrollo de sistemas de información considera la aplicación de una metodología como un factor sumamente trascendental para el éxito o fracaso de un proyecto.
- 3) Observamos que en efecto la aplicación de una metodología a un proyecto hace que este se desarrolle de manera más eficiente y organizada, ya que según los entrevistados dicen no saber de ningún proyecto fallido debido a la utilización de una metodología, siempre y cuando se haya implementado de la manera adecuada.
- 4) Comprobamos que las metodologías de desarrollo de software más utilizadas son las orientadas a objetos y las estructuradas.
- 5) Obtuvimos una hipótesis preliminar: "La carencia de conocimientos sobre aspectos importantes al elegir una metodología de desarrollo de software, que cubra las necesidades específicas del proyecto a desarrollar, origina un desarrollo de software carente de grado de predicibilidad y eficiencia, los tiempos y resultados del desarrollo no coinciden con lo estipulado, generando así inconformidad e insatisfacción del cliente".

Conclusiones del marco teórico

En el Marco teórico:

- 1) La lectura del libro “Ingeniería de Software” fue indispensable para el desarrollo de nuestra tesis, ya que la temática que aborda va desde conceptos básicos de software, pasando por la metodología estructurada hasta llegar a la orientada a objetos. De hecho nuestra tesis esta basada en gran parte en conceptos de este libro.
- 2) La lectura del libro “Análisis y Diseño Orientado a Objetos” fue la base que empleamos para el tema, ya que trata a profundidad tópicos relacionados con el software, sistemas complejos y métodos. Menciona de diseño de sistemas complejos, especificando su significado y la importancia de construir un modelo, así como, los elementos de diseño de software y los modelos de desarrollo orientado a objetos.
- 3) Con la lectura del libro “Desarrollo y Gestión de Proyectos Informáticos” abordamos conceptos como desarrollo rápido, un tópico relativamente nuevo que se esta llevando acabo. En el libro se menciona que la velocidad con que desarrollemos cualquier programa concreto dependerá de la proporción de métodos eficaces y de métodos orientados que seleccionemos, para así lograr la velocidad en el desarrollo.
- 4) La lectura de otros libros nos permitió ampliar este estudio con diversos conceptos y temas relacionados. Estudiamos tópicos tratados en algunas tesis relacionadas con el tema, encontramos mayoritariamente del tema orientado a objetos.
- 5) Después de documentarnos nos dimos cuenta que no existe un estudio o tratado donde se clasifiquen los proyectos de acuerdo a su tipo para elegir la metodología con que se desarrollará. Por ello la presente investigación se enfocó en las principales metodologías que se utilizan actualmente en el mercado laboral, para aportar esta información a las empresas que desarrollan software, esperando que sea de utilidad.
- 6) Al realizar la investigación encontramos un gran número de fuentes de información acerca de las metodologías estructuradas y orientadas a objetos. Sin embargo notamos una menor cantidad de documentación en las metodologías de más reciente utilización como son las ágiles, solamente encontramos algunos libros en inglés y algunas ligas en Internet.

Conclusiones del marco conceptual

El Marco conceptual:

1) Es muy interesante ver como ha sido la evolución de cada tipo de metodología, un ejemplo es la estructurada, la cuál además es la base de muchas metodologías de desarrollo de software. La evolución de esta metodología ha ido desde las orientadas a datos no jerárquicos, orientadas a datos jerárquicos, orientadas a procesos hasta las mixtas.

2) Las metodologías orientadas a objetos (OO) se dividen en: revolucionarias o puras, sintetistas o evolutivas y además con una segunda generación como lo es RUP la unificación de dos de las metodologías más importantes (BOOCH & RUMBAUGH).

3) De la metodología de prototipo no encontramos metodologías específicas, sino varios tipos de prototipos: prototipos de viabilidad, prototipos de necesidades, prototipos de diseño, prototipos de implantación.

4) El modelo ágil de desarrollo ha originado nuevas metodologías que están siendo utilizadas para hacer más eficiente el desarrollo de software, tal es el caso de *Programación eXtrema*, el movimiento del desarrollo ágil de software es una iniciativa que agrupa una serie de metodologías que se basan en la adaptabilidad ante el cambio como medio para aumentar las posibilidades de éxito de un proyecto. En general los procesos ágiles se centran en las personas; en su comunicación directa y sus habilidades en vez de en procesos muy formales.

5) Al finalizar este capítulo aprendimos más sobre cada una de las metodologías de desarrollo de software existentes y sobre el contexto de desarrollar software con la mejor calidad posible. Por lo que podemos afirmar que entre más conocimientos se tengan sobre las metodologías de desarrollo de software, será más fácil y mejorará la elección de cada una de ellas para un proyecto específico de desarrollo de software.

Conclusiones del marco metodológico

En el Marco metodológico:

- 1) Que existe conciencia de lo importante que es emplear una metodología de desarrollo de software adecuada al tipo de proyecto, ya que esto implica crear productos de calidad optimizando recursos para satisfacer las necesidades de los clientes y usuarios.
- 2) Con la recopilación de la información del cuestionario final obtuvimos más conocimiento sobre diferentes variables, que intervienen en el uso de las metodologías, como son el que no existe una metodología universal, existen muchas, las cuales se pueden adaptar a cierto tipo de proyectos; también el que podemos retomar de diferentes metodologías lo que nos sea útil y adaptarla al tipo de proyecto que realizamos, lo más importante no es cierto tipo de metodología, sino tener una metodología adecuada y llevarla a cabo.
- 3) Con esta información pudimos aprobar nuestra hipótesis, ya que los entrevistados identificaron los beneficios más importantes por los que se utilizan las metodologías para el desarrollo de software como son: la planificación y el control del proyecto, el proceso estándar para una comunicación efectiva, el involucramiento del cliente y para tener la documentación del proyecto.

Conclusiones del marco instrumental

En el Marco instrumental:

- 1) Hicimos una página Web con lo cual nos dimos cuenta de que existen muy pocos sitios referentes a las Metodologías de Desarrollo de Software.
- 2) Propusimos una materia optativa sobre nuestro tema porque observamos que por la temática que aborda no podría ser una materia básica, sino que requiere de conocimientos básicos.

Conclusiones generales

- 1) En la presente investigación identificamos las metodologías de desarrollo de software más conocidas y utilizadas en la actualidad, confirmamos que el empleo de las metodologías de desarrollo de software ayuda a hacer eficiente el desarrollo de sistemas.
- 2) Aprendimos que existen infinidad de metodologías, de las cuales podremos optar por alguna dependiendo el tipo y tamaño de proyecto, ajustándola a nuestras necesidades específicas.
- 3) Gracias a esta investigación creemos que es muy importante que los profesionistas en Informática tengan vasto conocimiento sobre las Metodologías de Desarrollo de Software, porque así aseguran el correcto uso de estas, la identificación de la metodología idónea para cada tipo de proyecto o el retomar de algunas metodologías lo que les es útil, en el último de los casos crear una metodología propia.
- 4) Identificamos, que además de las Metodologías de Desarrollo de Software hoy en día han surgido una serie de herramientas, técnicas y modelos que facilitan a las organizaciones, encargadas de las tecnologías de la información, generar productos que cumplan las expectativas del cliente e incluso las rebasen, herramientas que prometen ser la solución a los problemas de calidad, costo y tiempos de desarrollo; de éstas podemos mencionar a los (modelos de calidad) como la norma ISO 9000-2000, el modelo Capability Maturity Model (CMM) del Software Engineerig Institute (SEI). Así como el Programa para el Desarrollo de la Industria de Software (PROSOFT).

ANEXOS

A.- Cuestionario para obtener conocimiento empírico

1	<p>Pregunta: ¿Qué metodologías para el desarrollo de software conoce? Escriba el inciso y las metodologías que conoce del mismo.</p> <p>a) Metodología estructurada: métodos que siguen el ciclo de vida tradicional o clásico. (YOURDON, SSADM (Structured Systems Analysis and Design), MÉTRICA).</p> <p>b) Metodologías de prototipo: proyectos basados alrededor de técnicas de prototipo. Es conveniente el ciclo de vida en espiral o prototipo evolutivo.</p> <p>c) Metodologías ágiles de desarrollo: híbridas, lo más importante es la velocidad en el desarrollo de sistemas. (Programación extrema).</p> <p>d) Paradigmas de diseño orientado a objetos: se alejan del ciclo de vida tradicional. (OMT (Object Modeling Technique), RUP (El Proceso Unificado de Rational)).</p> <p>e) Métodos de diseño formales: basados en matemáticas.</p> <p>f) Otras.</p> <p>Objetivo: Conocer las metodologías de desarrollo de software más utilizadas en la actualidad.</p> <p>Respuesta esperada: Algún inciso.</p>
2	<p>Pregunta: ¿Cree usted que llevar a cabo el desarrollo de software mediante una metodología impone un proceso disciplinado?</p> <p>a) Sí</p> <p>b) No</p> <p>c) No se</p> <p>Objetivo: Comprobar que una metodología implica una disciplina en el desarrollo de software.</p> <p>Respuesta esperada: Sí</p>

3	<p>Pregunta: ¿Considera que una metodología hace que el desarrollo de software sea más predecible y eficiente?</p> <p>a) Sí b) No c) No se</p> <hr/> <p>Objetivo: Identificar si las metodologías hacen que el desarrollo de software sea más predecible y eficiente.</p> <hr/> <p>Respuesta esperada: Sí.</p>
4	<p>Pregunta: ¿Cuál cree que sea el mejor tipo de metodología para el desarrollo de software?</p> <p>a) Estructurada b) Prototipo c) Ágiles d) Orientada a objetos e) Formales f) Otras, ¿Mencione cuáles?</p> <hr/> <p>Objetivo: Identificar cuál es la mejor metodología de desarrollo de software.</p> <hr/> <p>Respuesta esperada: Algún inciso.</p>
5	<p>Pregunta: ¿Cuál considera que sea el factor más importante para la elección de una metodología para el desarrollo de software?</p> <p>a) Cubrir necesidades específicas. b) Cumplir con el tiempo y costos establecidos. c) Tipo y tamaño del proyecto. d) Que el sistema haga lo que tenga que hacer. e) No se.</p> <hr/> <p>Objetivo: Identificar que factores se deben tomar en cuenta para la elección de una metodología, de acuerdo a esto se pueden clasificar.</p> <hr/> <p>Respuesta esperada: Inciso a, b, c, d.</p>

6	<p>Pregunta: ¿Cree usted que existe una metodología para el desarrollo de software eficaz para cada tipo de proyectos?</p> <p>a) Sí b) No c) No se</p> <p>Objetivo: Identificar si determinado proyecto involucra una metodología que se integre a sus necesidades específicas.</p> <p>Respuesta esperada: Sí</p>
7	<p>Pregunta: ¿Considera que se puede prescindir del análisis detallado en proyectos pequeños?</p> <p>a) Sí b) No c) No se</p> <p>Objetivo: Comprobar si efectivamente, no hace falta un análisis profundo en proyectos pequeños.</p> <p>Respuesta esperada: Sí</p>
8	<p>Pregunta: ¿Cree usted que es necesario el análisis detallado en proyectos grandes o de gran envergadura?</p> <p>a) Sí b) No c) No se</p> <p>Objetivo: Comprobar que en proyectos grandes es necesario un análisis profundo y detallado.</p> <p>Respuesta esperada: Sí</p>

9	<p>Pregunta: ¿Qué opina acerca del desenvolvimiento de la metodología estructurada?</p> <p>a) Excelente b) Bueno c) Regular d) Malo e) No la conozco</p> <hr/> <p>Objetivo: Conocer la importancia que tiene la metodologías estructuradas.</p> <hr/> <p>Respuesta esperada: Es buena, cuando se requieren de esas metodologías en específico.</p>
10	<p>Pregunta: ¿Qué opina acerca del desenvolvimiento de la metodología orientada a objetos?</p> <p>a) Excelente b) Bueno c) Regular d) Malo e) No la conozco</p> <hr/> <p>Objetivo: Conocer la importancia que tienen las metodologías orientadas a objetos.</p> <hr/> <p>Respuesta esperada: Funciona bastante bien en proyectos de gran envergadura en los que se requiere de un análisis arduo y profundo.</p>
11	<p>Pregunta: ¿Cree usted que las metodologías estructuradas retarden el proceso de desarrollo?</p> <p>a) Sí b) No c) No se</p> <hr/> <p>Objetivo: Comprobar si las metodologías de análisis estructurado exigen demasiado tiempo y esfuerzo en documentación que después no nos servirá.</p> <hr/> <p>Respuesta esperada: A veces exigen demasiado, pero es necesario que se lleve toda esa documentación y análisis en algunos proyectos.</p>

12	<p>Pregunta: ¿Cree usted que las metodologías orientadas a objetos retarden el proceso de desarrollo?</p> <p>a) Sí b) No c) No se</p>
	<p>Objetivo: Identificar si las metodologías orientadas a objetos exigen demasiado tiempo y esfuerzo en análisis y documentación, que después no será de gran utilidad.</p> <p>Respuesta esperada: Es totalmente necesario ese análisis y documentación en proyectos de gran envergadura.</p>
13	<p>Pregunta: ¿Conoce algún proyecto de desarrollo de sistemas que haya fracasado por causa de una metodología de desarrollo software?</p> <p>a) Sí b) No c) No se</p>
14	<p>Pregunta: ¿Qué opina acerca del desenvolvimiento de la metodología ágil o evolutiva?</p> <p>a) Excelente b) Bueno c) Regular d) Malo e) No la conozco</p>
	<p>Objetivo: Conocer la importancia de las metodologías ágiles.</p> <p>Respuesta esperada: Ha estado funcionando muy bien en proyectos pequeños en los que no se requiere de mucho análisis y no es pesado estar haciendo pequeños prototipos para que el cliente los analice y verifique que cumplan sus expectativas.</p>

B.- Recopilación del cuestionario

Nombre	Iniciales
Esteban Aguilar Landa	EAL
José Alfredo Escorza Escorza	JAEE
Erick Alvarado Ramírez	EAR
Karla Lorena Pérez Islas	KLPI
Guillermo Reyes Cordero	GRC
Isabel Vilchis González	IVG

Pregunta	EAL	JAEE	EAR	KLPI	GRC	IVG
1.- ¿Qué metodologías para el desarrollo de software conoce?	a) Yourdon, SSADM.	a) Metodología estructurada, b) Metodologías de prototipo.	a) Metodología estructurada, d) RUP.	a) Yourdon, d) RUP.	a) Yourdon, SSADM b) Metodologías de prototipo, d) RUP.	a) Yourdon, b) Metodologías de prototipo, d) RUP.
2.- ¿Cree usted que llevar acabo el desarrollo de software mediante una metodología impone un proceso disciplinado?	a) Si	a) Si	a) Sí, porque el éxito del desarrollo de un nuevo sistema depende en gran medida de saber exactamente cual es el problema.	a) Si	a) Sí, ya que una metodología implica una serie de pasos a seguir.	a) Si
3.- ¿Considera que una metodología hace que el desarrollo de software sea más predecible y eficiente?	a) Si	a) Si	a) Sí, ya que las metodologías se supone estudian todas las variables que el desarrollo del software implica.	a) Si	a) Si esa metodología implica un análisis profundo de los requerimientos, claro que si.	a) Si

4.- ¿Cuál cree que sea el mejor tipo de metodología para el desarrollo de software?	a) Estructurada.	a) Estructurada.	f) Eso depende en gran medida del tipo de sistema a cubrir.	d) Orientada a objetos.	f) Depende del sistema que se va a desarrollar.	f) Yo creo que depende del proyecto a desarrollar.
5.- ¿Cuál considera que sea el factor más importante para la elección de una metodología para el desarrollo de software?	a) Cubrir necesidades específicas.	b) Cumplir con el tiempo y costos establecidos. c) Tipo y tamaño del proyecto.	c) El tipo de sistema a desarrollar.	a) Cubrir necesidades específicas.	a) Cubrir necesidades específicas. c) Tipo y tamaño del proyecto.	d) Que el sistema haga lo que tenga que hacer.
6.- ¿Cree usted que existe una metodología para el desarrollo de software eficaz para cada tipo de proyectos?	a) Sí	b) No	a) Sí	b) No	a) Sí	a) Sí
7.- ¿Considera que se puede prescindir del análisis detallado en proyectos pequeños?	a) Sí	b) No	a) Sí	b) No	a) Sí, ya que sólo retardaría la entrega del proyecto.	b) No
8.- ¿Cree usted que es necesario el análisis detallado en proyectos grandes o de gran envergadura?	a) Sí	a) Sí	a) Sí, ya que el buen funcionamiento de un sistema complejo depende del análisis del problema.	a) Sí	a) Sí, es imprescindible de no haberlo se crearía el caos total.	a) Sí
9.- ¿Qué opina acerca del desenvolvimiento de las metodologías estructuradas?	b) Bueno	b) Bueno	b) Para la mayoría de los proyectos en México es bueno y apropiado su uso.	c) Regular	a) Es muy bueno y es básico para el aprendizaje ya que en este tipo de metodología todo se va formando	a) Excelente

					paso a paso.	
10.- ¿Qué opina acerca del desenvolvimiento de las metodologías orientadas a objetos?	e) No la conozco	b) Bueno	e) No la conozco	a) Excelente	b) Es bueno, empieza a haber un auge en las actuales empresas mexicanas.	b) Bueno
11.- ¿Cree usted que las metodologías estructuradas retarden el proceso de desarrollo?	b) No	b) No	a) Sí se es muy rígido con la metodología si se puede llegar a ser burocrático.	c) No sé	a) Sólo si se aplica de tal manera que no se acepten ningún tipo de concesiones.	b) No
12.- ¿Cree usted que las metodologías orientadas a objetos retarden el proceso de desarrollo?	c) No sé	b) No	c) No sé	b) No	b) No	b) No
13.- ¿Conoce algún proyecto de desarrollo de sistemas que haya fracasado por causa de una metodología de desarrollo software?	b) No	b) No	b) No tanto por la metodología, más bien por un mal análisis si he conocido malos proyectos.	b) No	b) No	b) No
14.- ¿Qué opina acerca del desenvolvimiento de la metodología ágil o evolutiva?	b) Bueno	e) No la conozco	e) No la conozco	e) No la conozco	e) No la conozco	e) No la conozco

C.- Cuestionario para obtener opiniones profesionales

1	<p>Pregunta: ¿Qué metodologías para el desarrollo de software conoce? Escriba el inciso y las metodologías que conoce del mismo.</p> <p>a) Metodología estructurada: métodos que siguen el ciclo de vida tradicional o clásico. (YOURDON, SSADM (Structured Systems Analysis and Design), MÉTRICA).</p> <p>b) Metodologías de prototipo: proyectos basados alrededor de técnicas de prototipo. Es conveniente el ciclo de vida en espiral o prototipo evolutivo.</p> <p>c) Metodologías ágiles de desarrollo: híbridas, lo más importante es la velocidad en el desarrollo de sistemas. (Programación extrema).</p> <p>d) Paradigmas de diseño orientado a objetos: se alejan del ciclo de vida tradicional. (OMT (Object Modeling Technique), RUP (El Proceso Unificado de Rational)).</p> <p>e) Métodos de diseño formales: basados en matemáticas.</p> <p>f) Otras.</p>
	<p>Objetivo: Conocer las metodologías de desarrollo de software más utilizadas en la actualidad.</p> <p>Respuesta esperada: Inciso a, c ó d.</p>
2	<p>Pregunta: ¿Utiliza alguna metodología de desarrollo de software?, ¿Cuál es?</p> <p>Objetivo: Conocer cuales son las metodologías de desarrollo de software más utilizadas.</p> <p>Respuesta esperada: Cualquier metodología.</p>
3	<p>Pregunta: ¿Específicamente para qué usa esta metodología para el desarrollo de software?</p> <p>a) Planificación y control del proyecto.</p> <p>b) Mejores aplicaciones.</p> <p>c) Proceso estándar para una comunicación efectiva.</p> <p>d) Para tener la documentación del proyecto.</p> <p>e) No tengo una clara idea de porque se utiliza esta metodología.</p> <p>f) Otras, escriba cuales:</p> <p>Objetivo: Identificar la utilización de las metodologías de desarrollo de software.</p> <p>Respuesta esperada: inciso a, b ó d.</p>

4	<p>Pregunta: ¿Aplica formalmente todos los elementos de la notación de esta metodología?</p> <p>a) Sí, ¿Por qué? b) No, ¿Por qué? c) No se</p> <hr/> <p>Objetivo: Comprobar que las metodologías pueden ser muy flexibles, además de que cada quien puede crear una metodología propia que se adapte mejor a sus necesidades.</p> <hr/> <p>Respuesta esperada: No, sólo utilizo los elementos que me resultan más prácticos.</p>
5	<p>Pregunta: ¿Considera que se han obtenido beneficios en el desarrollo de sistemas a partir del uso de esta metodología?</p> <p>a) Sí, ¿Por qué? b) No, ¿Por qué? c) No se</p> <hr/> <p>Objetivo: Ver que beneficios ha aportado el uso de esta metodología.</p> <hr/> <p>Respuesta esperada: Sí</p>
6	<p>Pregunta: ¿Cuál considera que es la razón por la que han obtenido o no beneficios?</p> <p>a) Planificación y control. b) Existencia de reglas predefinidas. c) El uso de la documentación para lograr una comunicación efectiva. d) Especificar claramente los requerimientos, no permitir flexibilidad en los cambios e) No se ha obtenido ningún beneficio. f) Otros, escriba cuales son:</p> <hr/> <p>Objetivo: Identificar porque se obtiene ciertos beneficios con el uso de esta metodología.</p> <hr/> <p>Respuesta esperada: Inciso a, b ó c.</p>

7	<p>Pregunta: ¿Ha sido complicado incorporar esta metodología?</p> <p>a) Sí, ¿Por qué? b) No, ¿Por qué? c) No se</p> <p>Objetivo: Identificar el grado de dificultad que implica esta metodología.</p> <p>Respuesta esperada: No</p>
8	<p>Pregunta: ¿Considera que el uso de esta metodología ha facilitado el desarrollo de sistemas?</p> <p>a) Sí, ¿Por qué? b) No, ¿Por qué? c) No se</p> <p>Objetivo: Comprobar que el uso de las metodologías facilita el desarrollo de software.</p> <p>Respuesta esperada: Sí, efectivamente.</p>
9	<p>Pregunta: ¿Es conocida y comprendida la metodología de desarrollo de software que utiliza en su organización?</p> <p>a) Sí, ¿Por qué? b) No, ¿Por qué? c) No se</p> <p>Objetivo: Verificar si la metodología es comprensible para los que la implementan.</p> <p>Respuesta esperada: Sí.</p>

10	<p>Pregunta: ¿Qué tipo de proyectos desarrolla con esta metodología para el desarrollo de software?</p> <p>a) Sistemas complejos, sistemas grandes b) Sistemas medianos, de fácil comprensión c) Sistemas pequeños d) Otros, diga cuales son:</p> <hr/> <p>Objetivo: Identificar las metodologías que involucran determinado tipo de proyectos.</p> <hr/> <p>Respuesta esperada: a, b ó c.</p>
11	<p>Pregunta: ¿Cree usted que la metodología que utiliza es la mejor?, ¿Por qué?</p> <p>a) Sí, ¿Por qué? b) No, ¿Por qué? c) No se</p> <hr/> <p>Objetivo: Identificar si las metodologías se adaptan a las necesidades.</p> <hr/> <p>Respuesta esperada: Sí.</p>
12	<p>Pregunta: ¿Cuáles son las ventajas de usar la metodología que usted utiliza?</p> <p>a) Seguimiento ordenado y completo b) Mejor organización c) Madurez y eficiencia en el desarrollo d) Se presenta al usuario una versión trabajando del producto final al principio del proceso de desarrollo. e) Retroalimentación con el cliente f) No hay ventajas g) Otras, diga cuales son:</p> <hr/> <p>Objetivo: Identificar las ventajas de las metodologías de desarrollo de software.</p> <hr/> <p>Respuesta esperada: Cualquier inciso menos el f.</p>

13	<p>Pregunta: ¿Cuáles son las desventajas de usar la metodología que usted utiliza?</p> <ul style="list-style-type: none">a) Dificultad para especificar claramente los requerimientosb) No permite flexibilidad en los cambiosc) Requiere conocimientos y experiencia altamente especializadosd) No proporciona resultados tangibles en forma de software antese) Saber cuando dejar de refinar y cuando consentir con el sistemaf) Imposibilidad de conocer al comienzo del proyecto el tiempo y costog) No hay desventajash) Otras, diga cuales son: <p>Objetivo: Identificar las desventajas de las metodologías de desarrollo de software.</p> <p>Respuesta esperada: Cualquier inciso menos la g.</p>
14	<p>Pregunta: ¿Conoce algún proyecto de desarrollo de sistemas que haya fracasado por causa de una metodología de desarrollo software?</p> <ul style="list-style-type: none">a) Sí, ¿Por qué?b) No, ¿Por qué?c) No se <p>Objetivo: Identificar la importancia que tiene las metodologías en el desarrollo de software.</p> <p>Respuesta esperada: Sí.</p>

D.- Recopilación del cuestionario

Nombre	Iniciales
Karla Arias Mondragón	KAM
Luis Carlos Infante Cisneros	LCIC
Cecilia Herrera Marín	CHM
Anabel Martínez Colín	AMC
Omar Eduardo Ortiz Garza	OEOG
Elsa Patricia Urtecho Altamirano	EPUA

PREGUNTA	KAM	LCIC	CHM	AMC	OEOG	EPUA
1.- ¿Qué metodologías para el desarrollo de software conoce?	a) Metodología estructurada b) Metodologías de prototipo. d) RUP.	a) Yourdon, b) Metodologías de prototipo, c) RUP.	a) Metodología estructurada.	a) Métrica 3, clásica, d) RUP, CASE.	a) Metodología estructurada, b) Metodologías de prototipo.	a) Yourdon, SSADM, MÉTRICA.
2.- ¿Utiliza alguna metodología de desarrollo de software?, ¿Cuál es?	Sí, la de Yourdon por el momento. En un futuro no muy lejano la de RUP.	RUP	Estructurada. Propia de la empresa.	Una combinación entre la Clásica y la Métrica 3.	No, usamos herramientas de diversas metodologías.	No estoy utilizando actualmente una metodología.
3.- ¿Específicamente para qué usa esta metodología para el desarrollo de software?	a) Planificación y control del proyecto, b) Mejores aplicaciones, c) Proceso estándar para una comunicación efectiva, d) Para tener la documentación del	f) Para distribuir tareas entre diversos analistas y programadores de manera más eficiente.	b) Mejores aplicaciones.	f) Para llevar un orden en el desarrollo, y dar un seguimiento adecuado al mismo así como un buen término de cada actividad.	a) Planificación y control del proyecto, b) Mejores aplicaciones, c) Proceso estándar para una comunicación efectiva, d) Para tener la documentación del	a) Planificación y control del proyecto, b) Mejores aplicaciones.

	proyecto. f) Además para crear una propia cultura dentro de la empresa y crear disciplina en el área.				proyecto.	
4.- ¿Aplica formalmente todos los elementos de la notación de esta metodología?	b) No, porque en las metodologías intervienen variedad de personas, a lo mejor una sí aplica notación pero otras no, también depende de las políticas de la empresa y la formación de cada profesional.	b) No.	a) Sí, por que es un lenguaje ya establecido.	b) No todos por ello es una combinación de dos metodologías.	a) Sí, para que sea estándar para todos.	b) No, porque no aplican a la aplicación o por informalidad.
5.- ¿Considera que se han obtenido beneficios en el desarrollo de sistemas a partir del uso de esta metodología?	a) Sí, pues un mejor control, documentación de inicio a fin del proyecto.	a) Sí	a) Sí, cuando se sigue un orden y se cumplen y/o exigen los entregables.	a) Sí	a) Sí, reducción en tiempo de desarrollo en fallas y en mantenimiento	a) Sí, mejor control y organización del sistema en desarrollo.
6.- ¿Cuál considera que es la razón por la que han obtenido o no beneficios?	a) Planificación y control.	f) Mejorar el nivel de comunicación entre los que participan en el desarrollo de sistemas.	d) Se especifican claramente los requerimientos, teniendo cierta flexibilidad en los cambios, cuando estos son drásticos queda documentado y se planean	f) Por el hecho de realizar un buen análisis y desarrollo del software.	a) Planificación y control, d) Especificar claramente los requerimientos, no permitir flexibilidad en los cambios.	a) Planificación y control, b) Existencia de reglas predefinidas, e) No se ha obtenido ningún beneficio.

			nuevas fechas de entrega.			
7.- ¿Ha sido complicado incorporar esta metodología?	b) No porque es ya parte de la vida de sistemas y además en todas partes ya se exige tener una metodología de desarrollo.	b) No	b) No, ya la usan.	a) Sí, en algunas ocasiones, por cuestiones de tiempo.	a) Sí, a las personas les cuesta trabajo seguir reglas, creen que el desarrollo es inspiración.	a) Sí, por la idiosincrasia de los directivos.
8.- ¿Considera que el uso de esta metodología ha facilitado el desarrollo de sistemas?	a) Sí porque permite tener ciertos criterios y parámetros para iniciar y hasta implementar un sistema, pasando por las diversas etapas del mismo.	a) Sí	a) Sí, queda más clara la definición de requerimientos tanto para los usuarios como para los desarrolladores.	a) En mi caso particular sí.	a) Sí, todos los miembros del equipo hablan el mismo idioma.	a) Sí, porque existe una manera para manejar la información.
9.- ¿Es conocida y comprendida la metodología de desarrollo de software que utiliza en su organización?	a) Sí	a) Sí	a) Sí	a) Sí	a) Sí, se capacita en las herramientas usadas para que todos entiendan los proyectos.	Es variable, porque no todos utilizan la misma metodología.
10.- ¿Qué tipo de proyectos desarrolla con esta metodología para el desarrollo de software?	a) Sistemas complejos, sistemas grandes, es decir, básicamente los sistemas que a nivel operacional, soportan la operación de la empresa.	d) Software financiero y administrativo.	c) Sistemas pequeños.	c) Desarrollo de sistemas pequeños y medianos, así como el mantenimiento de los mismos.	a) Sistemas complejos, sistemas grandes, b) Sistemas medianos, de fácil comprensión.	a) Sistemas complejos, sistemas grandes, b) Sistemas medianos, de fácil comprensión, c) Sistemas pequeños.
11.- ¿Cree usted que la metodología que utiliza es la mejor?	c) No sé, pienso que va en función de las necesidades de sistemas	a) Sí porque gracias a la metodología se han obtenido buenos	a) Cualquier metodología estructurada considero que es buena y que ayuda a	a) Sí, Porque es la que considero más completa.	c) No sé	b) No

	de la empresa.	la resultados.	la organización.			
12.- ¿Cuáles son las ventajas de usar la metodología que usted utiliza?	a) Seguimiento ordenado y completo, e) Retroalimentación con el cliente.	b) Mejor organización del proyecto que se traduce en buenos resultados para el cliente.	b) Mejor organización, e) Retroalimentación con el cliente.	a) Se lleva un seguimiento ordenado y completo.	a) Seguimiento ordenado y completo.	a) Seguimiento ordenado y completo, b) Mejor organización, c) Madurez y eficiencia en el desarrollo.
13.- ¿Cuáles son las desventajas de usar la metodología que usted utiliza?	e) Saber cuando dejar de refinar y cuando consentir con el sistema.	h) Retarda un poco el proceso de desarrollo la elaboración de diagramas y demás requisitos de la metodología.	g) No hay desventajas.	h) El grado de detalle.	c) Requiere conocimientos y experiencia altamente especializados.	b) No permite flexibilidad en los cambios, c) Requiere conocimientos y experiencia altamente especializados.
14.- ¿Conoce algún proyecto de desarrollo de sistemas que haya fracasado por causa de una metodología de desarrollo software?	b) No	b) No	b) No	a) Si pero no creo que al 100% se deba a la metodología sino al hecho de que no se implemento de manera adecuada.	a) Si, si el proyecto es de mediano a grande y no se siguen reglas el proyecto se vuelve difícil de manejar y entra en caos.	b) No, porque se maneja su control de alguna manera.

E.- Cuestionario final



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN



Cuestionario aplicado por las alumnas **Ma. Guadalupe Bautista Zaragoza** y **Margarita Garibay Tierradentro** con número de cuenta 099517557 y 095289841 respectivamente, pasantes de la Licenciatura en Informática de la Facultad de Contaduría y Administración.

Objetivo: Comprobar la hipótesis definitiva en la investigación de la tesis denominada "Metodologías de desarrollo de software como una herramienta para el aseguramiento de la calidad".

Instrucciones: Conteste las siguientes preguntas, marque con una "X" su respuesta dentro del paréntesis. Algunas preguntas son abiertas por favor explique su punto de vista.

Preguntas:

1.- ¿Considera que las metodologías para el desarrollo de software son un factor primordial para el éxito de un proyecto?

- a.- () Sí es muy importante.
- b.- () Es importante.
- c.- () No es importante.

2.- ¿Cuáles considera que sean los aspectos más importantes para la elección de una metodología para el desarrollo de software?

- a.- () Cubrir necesidades específicas.
- b.- () Cumplir con el tiempo y costos establecidos.
- c.- () Tipo y tamaño del proyecto.
- d.- () Que el sistema haga lo que tenga que hacer.
- e.- () No se.

3.- ¿Utiliza alguna metodología de desarrollo de software?

- a.- () Metodología estructurada: métodos que siguen el ciclo de vida tradicional o clásico. (YOURDON, SSADM, MÉTRICA).
- b.- () Metodologías de prototipo: proyectos basados alrededor de técnicas de prototipo. Es conveniente el ciclo de vida en espiral o prototipo evolutivo.
- c.- () Metodologías ágiles de desarrollo: híbridas, lo más importante es la velocidad en el desarrollo de sistemas. (Programación eXtrema).
- d.- () Paradigmas de diseño orientado a objetos: se alejan del ciclo de vida tradicional. (OMT, OOD, RUP).
- e.- () Otras. Por favor menciónelas:

4.- ¿Qué tipo de proyectos desarrolla con esta metodología para el desarrollo de software?

Tamaño:

- a.- Sistemas complejos, sistemas grandes.
 - b.- Sistemas medianos, de fácil comprensión.
 - c.- Sistemas pequeños.
 - d.- Otros. Por favor menciónelos:
-

Tipo:

- a.- Crítico.
- b.- Administrativo / gestión de información.
- c.- Comercial.

5.- ¿Por qué utiliza esta metodología de desarrollo de software?

6.- ¿Desde cuándo utiliza esta metodología de desarrollo de software?

7.- ¿Aplica formalmente todos los elementos de la notación de esta metodología?

- a.- Sí
- b.- No
- c.- No se

Explique:

8.- ¿Específicamente para qué usa esta metodología para el desarrollo de software?

- a.- Planificación y control del proyecto.
 - b.- Mejores aplicaciones.
 - c.- Proceso estándar para una comunicación efectiva.
 - d.- Para tener la documentación del proyecto.
 - e.- No tengo una clara idea de porque se utiliza esta metodología.
 - f.- Otras. Por favor menciónelas:
-

9.- ¿Cuáles son las ventajas de usar la metodología que usted utiliza?

- a.- Seguimiento ordenado y completo.
 - b.- Existencia de reglas predefinidas.
 - c.- Cobertura total del ciclo de vida.
 - d.- Verificaciones intermedias.
 - e.- Planificación y control.
 - f.- Comunicación efectiva
 - g.- Utilización sobre un abanico amplio de proyectos.
 - h.- Fácil formación.
 - i.- Herramientas CASE.
 - j.- Actividades que mejoren el proceso.
 - k.- Soporte al mantenimiento.
 - l.- Soporte de la reutilización de software.
 - m.- No hay ventajas.
 - n.- Otras. Por favor menciónelas:
-

10.- ¿Cuáles son las desventajas de usar la metodología que usted utiliza?

- a.- Dificultad para especificar claramente los requerimientos.
 - b.- No permite flexibilidad en los cambios.
 - c.- Requiere conocimientos y experiencia altamente especializados.
 - d.- No proporciona resultados tangibles en forma de software.
 - e.- Los usuarios de las metodologías los interpretan según su punto de vista.
 - f.- Saber cuándo dejar de refinar y cuándo consentir con el sistema.
 - g.- Imposibilidad de conocer al comienzo del proyecto el tiempo y costo.
 - h.- No hay desventajas.
 - i.- Otras. Por favor menciónelas:
-

11.- ¿Ha sido complicado incorporar esta metodología?

- a.- Sí
- b.- No
- c.- No se

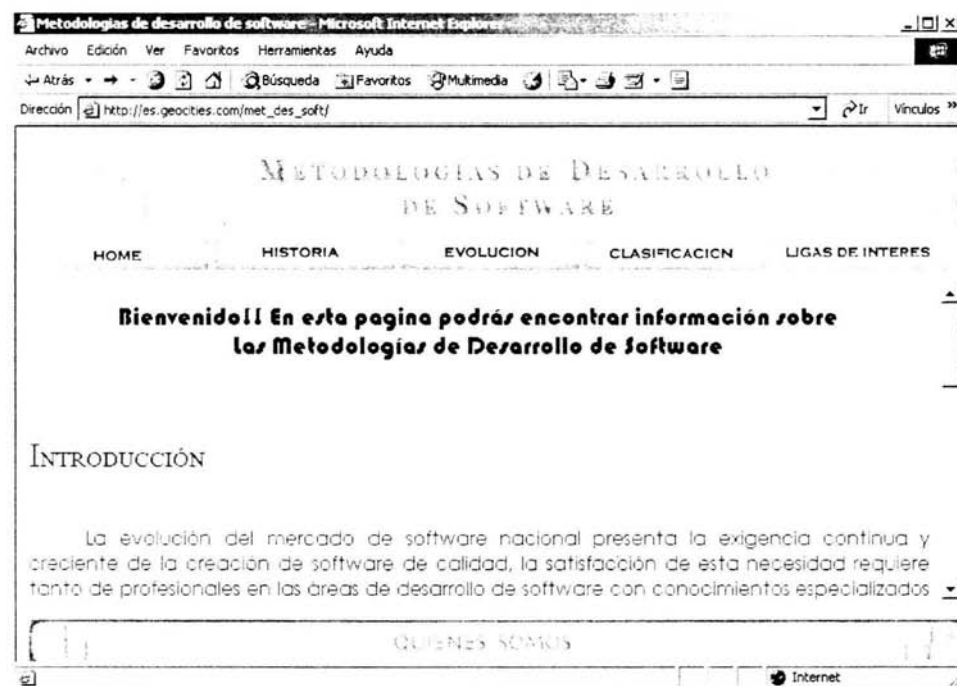
Explique: _____

12.- ¿Cree usted que existe una metodología para el desarrollo de software eficaz para cada tipo de proyectos?

- a.- Sí
- b.- No
- c.- No se

Comentarios: _____

F.- Página en Internet



The screenshot shows a Microsoft Internet Explorer browser window. The title bar reads "Metodologías de desarrollo de software - Microsoft Internet Explorer". The menu bar includes "Archivo", "Edición", "Ver", "Favoritos", "Herramientas", and "Ayuda". The address bar shows the URL "http://es.geocities.com/met_des_soft/". The website content features a header with the title "METODOLOGIAS DE DESARROLLO DE SOFTWARE" and a navigation menu with links for "HOME", "HISTORIA", "EVOLUCION", "CLASIFICACION", and "LIGAS DE INTERES". A central message reads: "Bienvenido!! En esta pagina podrás encontrar información sobre las Metodologías de Desarrollo de Software". Below this is a section titled "INTRODUCCIÓN" with a paragraph: "La evolución del mercado de software nacional presenta la exigencia continua y creciente de la creación de software de calidad, la satisfacción de esta necesidad requiere tanto de profesionales en las áreas de desarrollo de software con conocimientos especializados". At the bottom of the page, there is a section titled "QUIENES SOMOS". The browser's status bar at the bottom indicates "Internet".

The screenshot shows a Microsoft Internet Explorer browser window. The title bar reads 'Metodologías de desarrollo de software - Microsoft Internet Explorer'. The address bar contains the URL 'http://es.geocities.com/met_des_soft/'. The website content features a navigation menu with links for HOME, HISTORIA, EVOLUCIÓN, CLASIFICACION, and LIGAS DE INTERES. The main text discusses the purpose of the work, which is to complement a reference framework for a thesis on software development methodologies. It aims to identify existing methodologies, understand their approaches, and determine their scope within the software development lifecycle. The text concludes by stating that this knowledge base will help in choosing a methodology for system development.

METODOLOGIAS DE DESARROLLO DE SOFTWARE

HOME HISTORIA EVOLUCIÓN CLASIFICACION LIGAS DE INTERES

El propósito de este trabajo es complementar el Marco referencial de nuestra tesis que tiene como tema principal las **Metodologías de desarrollo de sistemas de información** para la licenciatura en Informática. En esta investigación se trata en primer lugar el de identificar los nombres de tantas metodologías existentes y en segundo lugar, conocer el enfoque de cada una, la manera en que definen, los conceptos básicos de ingeniería de software, de desarrollo de sistemas de información, qué parte del ciclo de vida abarcan cada una de las metodologías existentes o si abarcan todo el ciclo de vida. Qué recursos disponibles hay para utilizar una de estas metodologías, si existen libros disponibles, qué tipo de conocimientos necesitamos tener para poder utilizarlas, en qué lenguajes de programación se apoyan.

Y teniendo esta base de conocimientos, ahora si poder elegir una de ellas para utilizarla en el desarrollo de los sistemas en los que sea participe cualquier persona que lea este documento.

QUIENES SOMOS

G.- Carta de presentación para publicación de artículo

Asunto: Artículo para publicación

L. A. Gustavo Almaguer Pérez
Secretario de Divulgación y Fomento Editorial
Facultad de Contaduría y Administración

Presente

Debido a la creciente importancia que ha venido adquiriendo la tecnología de la información, actualmente es vista como una estrategia de negocio dentro del ámbito profesional teniendo como consecuencia un especial interés para el público en general, resultado de esto son los temas del área de informática como son el desarrollo de software y sistemas de información.

Como parte del *Marco Instrumental* de nuestra tesis de licenciatura en informática del tema "*Metodologías de desarrollo de software*", nos permitimos presentarle un breve artículo titulado "*Las metodologías de desarrollo de software son un factor primordial al construir sistemas informáticos...*", con la finalidad de que considere su publicación en la revista de "*Academia*" editada por la facultad de Contaduría y Administración.

Dicho artículo de opinión tienen como objetivo sembrar el interés por conocer más acerca del tema, esta dirigido a la comunidad en general y en especial a los estudiantes y/o egresados de la licenciatura en informática. (Se anexa de forma impresa y en disco flexible).

No dudando será de gran interés para la comunidad de la facultad de Contaduría de Administración, agradecemos su atención y quedamos a sus ordenes para cualquier comentario.

Ma. Guadalupe Bautista Zaragoza
No. de cuenta: 9951755-7
Egresada de la licenciatura en informática 1999-2002
Teléfono: 57431033
Correo electrónico: bz_guadalupe@yahoo.com.mx

Margarita Garibay Tierradentro
No. de cuenta: 9528984-1
Egresada de la licenciatura en informática 1998-2002
Teléfono: 56853816
Correo electrónico: gmarg74@hotmail.com

México, D. F., a 24 de junio de 2004.

*Recibido
Artículo para
publicar en
publicación
24 Junio 04*

H.- Artículo para publicación

Las metodologías de desarrollo de software son un factor primordial al construir sistemas informáticos...

Ma. Guadalupe Bautista Zaragoza
Margarita Garibay Tierradentro
Licenciatura en informática

Hoy en día el desarrollo de software es una actividad imprescindible en múltiples entidades organizacionales, esto tiene como consecuencia un amplio mercado que lucha porque el producto entregado a los clientes y/o usuarios sea una aplicación de calidad, que responda adecuadamente a sus necesidades específicas dentro del tiempo y costos requeridos.

¿Por qué las metodologías de desarrollo de software?

El desarrollo de sistemas pequeños, en la cual participan una o dos personas, es una tarea simple. Los cambios naturales que surgen durante el ciclo de desarrollo del sistema no producen una gran propagación de cambios en el sistema. Sin embargo, si el sistema es grande y en su desarrollo participan varios grupos de personas desarrollando una tarea específica, hay que tener en cuenta no sólo la comunicación con el usuario sino también la interrelación entre los distintos grupos de trabajo.

Algunos de los problemas comunes que los desarrolladores encuentran en la construcción de software de cierta complejidad son los siguientes:

- El dominio de aplicación no es conocido.
- La comunicación con el usuario.
- La comunicación con el grupo de desarrollo.
- La carencia de buena documentación.

Por esta razón, es necesario seguir una serie de pasos sistemáticos para que los diferentes grupos de desarrollo posean una buena comunicación. En las organizaciones dedicadas al desarrollo de sistemas informáticos actualmente es indispensable basar sus operaciones en un método para obtener un producto de calidad, para ello han existido múltiples formas que han querido llegar a este cometido, como lo es la ingeniería del software, dentro de la ingeniería del software existen habilidades técnicas y organizacionales, así como los

modelos del ciclo de vida; que en conjunto proporcionan una sana base que sustenta las metodologías de desarrollo de software ya existentes, y además da pauta para la creación de otras nuevas, con la combinación tanto de las metodologías como de los elementos que les dieron origen.

La importancia de las metodologías de desarrollo de software radica en que imponen un proceso disciplinado con el objetivo de hacer el trabajo más predecible, eficiente y planificado, permitiendo los siguientes aspectos:

- Seguimiento ordenado y completo.
- Existencia de reglas predefinidas.
- Cobertura total del ciclo de vida.
- Verificaciones intermedias.
- Planificación y control.
- Comunicación efectiva (Cercana e involucramiento con el cliente).
- Utilización sobre un abanico amplio de proyectos.
- Fácil formación.
- Herramientas CASE.
- Actividades que mejoren el proceso.
- Soporte al mantenimiento.
- Soporte de la reutilización de software.

Lo más importante es tener una metodología de desarrollo de software y seguirla, únicamente así se asegurará el resultado esperado, a veces en mayor tiempo o con mayor esfuerzo.

Aspectos importantes

Un factor determinante para incorporar una metodología de desarrollo de software es la gente, es decir, las personas que van a utilizarla, por ello se deduce que a medida que se tenga el personal adecuado (personal con conocimientos y experiencia), la metodología de desarrollo de software va a ser fácilmente incorporada y entendida por todos en la organización, y así se podrá optimizar su uso.

Es de vital importancia tener conocimientos sobre las metodologías de desarrollo de software para elegir la optima para cada proyecto, basándose en aspectos relevantes como son: tipo y tamaño de proyecto, así como necesidades específicas; esto nos permitirá crear un producto de calidad que satisfaga al cliente usuario.

En la actualidad existen infinidad de metodologías que nos permiten estar a la vanguardia en un medio donde los avances tecnológicos son acelerados. El uso de una metodología afecta directamente la calidad de vida del personal que trabaja en las actividades de

desarrollo de sistemas, ya que al tener una adecuada planeación de tiempo y recursos permite que la gente no pierda más tiempo y esfuerzo del que se requiere.

No hay una metodología universal; hay muchas que son muy efectivas y han sido ampliamente probadas, no obstante lo más recomendable es retomar lo mejor de éstas o de una en particular para intentar cubrir todas sus necesidades y aplicarla al contexto del tipo de proyecto y de la organización, es decir adaptarla a la situación particular. También se puede tratar de implementar su propia forma de hacer las cosas.

Conclusiones

Recomendamos tener especial cuidado al elegir una metodología de desarrollo de sistemas e identificamos los siguientes factores:

- Tener en cuenta lo importante que es emplear una metodología de desarrollo de software adecuada al tipo de proyecto a realizar, ya que esto implica crear productos de calidad optimizando recursos para satisfacer las necesidades de los clientes y usuarios.
- Dado que cada proyecto es único, no existe una metodología de desarrollo de software que se aplique al 100 % a todos los proyectos de una organización.
- La mayoría de los usuarios de las metodologías de desarrollo de software, no aplican formalmente todos los elementos de la notación de la metodología que utilizan debido a diversas razones como son: tiempo, costo, conocimientos de las personas y porque son modelos extranjeros que no aplican en su totalidad en empresas mexicanas.
- Una organización puede contar con una o más metodologías de desarrollo de software para ser utilizados dependiendo del tipo de proyecto.
- Se pueden omitir algunos elementos de las metodologías de desarrollo de software sin afectar el desarrollo de los sistemas informáticos.
- La metodología de desarrollo de software seleccionada tendrá influencia en el éxito del proyecto y en el tipo de decisiones que deberán tomarse.
- La gente que participa en el proyecto debe estar capacitada en el uso de las metodologías de desarrollo de software.
- Cualquier desviación a la metodología de desarrollo de software debe ser documentada y aprobada.

- Por lo anterior se deduce que no hay una metodología universal; hay muchas que son muy buenas y han sido ampliamente probadas no obstante lo más efectivo es retomar lo mejor de éstas o de una en particular y aplicarla al contexto de la organización y del tipo de proyecto.
- Sin una metodología difícilmente se logrará con éxito la construcción de un sistema.

I.- Carta de presentación para propuesta de asignatura

México, D. F., a 28 de junio de 2004.

Asunto: **Propuesta de asignatura**


Mtra. Graciela Bribiesca Correa
Jefa de la Licenciatura en Informática
Facultad de Contaduría y Administración


Presente

En el ámbito informático desde el punto de vista de desarrollo de software, lo primordial es la entrega de un producto de calidad que conlleve la satisfacción del cliente, es decir, el desarrollo de soluciones adecuadas que el licenciado en informática debe implementar.

Es por ello que es necesario ofrecer una mayor especialización en el área de las metodologías para desarrollo de software, ya que no existe una metodología universal y debido a la creciente complejidad que presentan los sistemas informáticos hay muchas que son muy eficientes y han sido ampliamente probadas, no obstante lo más efectivo es retomar lo mejor de éstas o de una en particular, y adaptar a la situación particular de cada proyecto. Para esto y como parte del "Marco Instrumental" de nuestra tesis de licenciatura en informática del tema "Metodologías de desarrollo de software", nos permitimos presentarle una propuesta para incorporar dentro del plan de estudios de la licenciatura en informática, una *asignatura optativa de Metodologías para desarrollo de software*, considerando que algunos aspectos de este tema son cubiertos en otras asignaturas, pero no con la especialización y el grado de especialización que lo amerita.

Esperando tome en consideración la presente propuesta para el beneficio principalmente de los egresados, anexo el desglose del contenido temático que sugerimos para la asignatura, agradecemos su atención y quedamos a sus órdenes para cualquier comentario al respecto




Ma. Guadalupe Bautista Zaragoza
No. de cuenta: 9951755-7
Egresada de la licenciatura en informática 1998-2001
Teléfono: 57431033
Correo electrónico: bz_guadalupe@yahoo.com.mx


Margarta Garibay Tierradentro
No. de cuenta: 9528984-1
Egresada de la licenciatura en informática 1998-2001
Teléfono: 56853816
Correo electrónico: gmarg74@hotmail.com

 FACULTAD DE CONTADURÍA
Y ADMINISTRACIÓN

29 JUN 2004

JEFATURA DE LA DIVISION
DE INFORMÁTICA



29/06/04

J.- Temario para materia optativa

Nombre:	Metodologías de desarrollo de software
Objetivo:	Los alumnos comprenderán el desarrollo de software como un proceso de ingeniería sistemático, encaminado a obtener productos que satisfagan las necesidades del cliente bajo restricciones de costes y plazos. Asimismo, los alumnos aprenderán a utilizar las metodologías, técnicas y estándares necesarios para definir la especificación de un sistema y participar en la gestión del desarrollo y el aseguramiento de la calidad de un producto software.
Prerrequisitos:	Haber cursado una asignatura relacionada con el proceso de desarrollo de sistemas, ingeniería de software.
Horas:	2 horas por clase, 2 clases por semana.
Contenido:	<p>I. Introducción</p> <ol style="list-style-type: none"> 1. Antecedentes de las metodologías de desarrollo de software. 2. Origen de las metodologías de desarrollo de software. 3. Visión histórica de las metodologías de desarrollo de software. 4. Fundamentos de las metodologías de desarrollo de software. <p>II. Evolución de las metodologías de desarrollo de software</p> <ol style="list-style-type: none"> 1. Desarrollo artesanal. 2. Desarrollo estructurado. 3. Desarrollo orientado a objetos. 4. Desarrollo ágil. <p>III. Metodologías de desarrollo de software.</p> <ol style="list-style-type: none"> 1. Estructurada (Yourdon). 2. Prototipos. 3. Orientada a Objetos (RUP). 4. Ágil (Programación extrema). <p>IV. Tendencias de las metodologías de desarrollo de software</p>
Bibliografía:	<p>Roger S. Pressman. Ingeniería de Software. Ed. Mc. Graw Hill. España 1998.</p> <p>Edward Yourdon. Análisis Estructurado Moderno. Ed. Prentice Hall. México 1997.</p> <p>James Martín, James I. Odell. Métodos Orientados a Objetos. Conceptos Fundamentales. Ed. Prentice Hall. México 1997.</p> <p>Steve McConnell. Desarrollo y Gestión de Proyectos Informáticos. Ed. Mc. Graw Hill. 1996.</p>

GLOSARIO ACRÓNIMOS Y SIGLAS

A

Abstracción: Es una descripción simplificada o especificación de un sistema que enfatiza algunos de los detalles o propiedades del sistema, mientras suprime otros.

Acoplamiento: Es una medida de la interdependencia relativa entre los módulos.

Ad hoc: (Expresión latina) Para un fin determinado; para este propósito particular. Por ejemplo, una comisión que se elige o se nombra para que cumpla con un objetivo definido.

Análisis de sistemas: Entender y especificar a detalle qué debe hacer un sistema de información. Especialidad profesional que implica determinar las necesidades de cómputo de una organización y el diseño de sistemas de cómputo para cubrir esas necesidades.

AOO (Análisis Orientado a Objetos).

B

Bases de datos (DataBase): Conjunto de datos relacionados que se almacenan de forma que se pueda acceder a ellos de manera sencilla, con la posibilidad de relacionarlos, ordenarlos en base a diferentes criterios, etc. Las bases de datos son uno de los grupos de aplicaciones de productividad personal más extendidos.

C

CAD/CAM (Computer-Aided Design/Computer-Aided Manufacturing): Diseño asistido por computadora y fabricación asistida por computadora. Es un software usado para diseñar productos tales como tableros de circuitos electrónicos en computadoras.

Calidad: Se refiere a las características mensurables: cosas que se pueden comparar con estándares conocidos como longitud, color, propiedades eléctricas, maleabilidad etc. Sin embargo, el software es en su gran extensión, como entidad intelectual, es más difícil de caracterizar que los objetos físicos.

CASE: Ingeniería de Software Asistida por Ordenador (Computer Aided Software Engineering).

Cohesión: Un módulo con cohesión realiza solamente una tarea dentro de un procedimiento software, requiriendo poca interacción con los procedimientos que se realizan en otras partes del programa. Dicho de otra manera un módulo con cohesión debería hacer idealmente una sola cosa.

Concurrencia: Es la propiedad que distingue un objeto que está activo de uno que no lo está.

D

DBMS (Data Base Management System): Sistema de manejo de base de datos. Es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de alguna tarea específica. Las funciones principales de un DBMS son: crear y organizar la base de datos, manejar los datos de acuerdo con las peticiones de los usuarios mantener la integridad de datos.

Diccionario de datos: Es un listado organizado de todos los elementos de datos que son pertinentes para el sistema, con definiciones precisas y rigurosas que permiten que el usuario y el analista del sistema tengan una misma comprensión de las entradas, salidas, de las componentes de los almacenes y también de los cálculos intermedios.

DRA (Rapid Application Development RAD): Es una forma de construir sistemas de información combinando el uso de herramientas CASE, prototipo y límites de tiempos cortos y fijos dentro de ciertos estándares de revisión y productividad confiables.

E

Encapsulación: En el proceso de ocultar todos los detalles de un objeto que no contribuyen a sus características esenciales.

Encapsulado: Técnica en la que la información a enviar se coloca en el área de datos de un paquete o cuadro. Puede encapsularse el paquete de un protocolo en otro.

Encapsulamiento: (En orientación a objetos) Se define como la propiedad de los objetos de permitir el acceso a su estado únicamente a través de su interfaz o de relaciones preestablecidas con otros objetos.

G

Grosso modo: (latinismo) De manera basta. La frase es equivalente a "sin rigurosa exactitud".

H

Hardware: Es la parte física de las computadoras, integrada por los elementos electrónicos, electromecánicos, cables y periféricos. Dispositivos electrónicos que proporcionan capacidad de cálculo y dispositivos electromecánicos, que proporcionan una función externa.

Herencia: (En orientación a objetos) Es compartir atributos y operaciones entre clases tomando como base una relación jerárquica. En términos generales se puede definir una clase que después se irá refinando sucesivamente para producir subclases. Todas las subclases poseen o heredan, todas y cada una de las propiedades de su superclase y añaden, además, sus propiedades exclusivas.

Herramientas CASE: Son herramientas computacionales que apoyan cada una de las distintas etapas del ciclo de desarrollo de un sistema de información. Conjunto de utilidades técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información.

Herramientas de la ingeniería del software: Proporcionan un soporte automático o semiautomático para el proceso y los métodos.

Herramientas de análisis y diseño: Capacitan al ingeniero del software para crear modelos del sistema que haya que construir. Los modelos contienen una representación de los datos, de la función y del comportamiento (en el nivel de análisis), así como las caracterizaciones del diseño de datos, arquitectura, procedimientos e interfaz.

Identidad: (En orientación a objetos) Los datos están cuantificados en entidades discretas y distinguibles denominadas objetos. Ejemplo una televisión, una bicicleta, un árbol. Los objetos pueden ser concretos, como un archivo en un sistema de archivos, o bien conceptuales como la política de planificación en un sistema operativo con multiprocesos. Cada objeto posee su propia identidad inherente. En otras palabras: dos objetos serán distintos aún cuando los valores de todos sus atributos (tales como el nombre y el tamaño) sean idénticos.

Información: Resultado de procesar u ordenar datos. Es uno de los recursos más importantes para la organización, y lo que vale la información es subjetivo, es decir, el valor lo damos nosotros.

Informática: Es la ciencia del tratamiento automático y racional de la información, considerada como soporte de los conocimientos y las comunicaciones. Se ocupa del análisis de la información vista como un recurso para la organización. Esto incluye las definiciones, usos, valor y distribución de la información.

Inherencia: Se le da este nombre a la compenetración íntima entre sujeto y predicado, o entre determinante y determinado. Indica «ya que una cosa (el sujeto) pertenece al género designado por el atributo (*la tierra es un planeta*), ya que posee la cualidad designada por este atributo, propiedad que puede ser cualidad constante (*la tierra es redonda*) o un accidente, estado o acción (*la tierra gira*)»

Interface de software: Son los lenguajes códigos y mensajes que utilizan los programas para comunicarse unos con otros, tal como en un programa de aplicación y el sistema operativo.

Interface de usuario: Son los teclados, ratones, diálogos, lenguajes de comando y menús empleados para la comunicación entre el usuario y la computadora.

IS: (siglas inglesas correspondientes a Information Systems) Sistemas de información

ISO 9000: Una serie de normas (9000, 9001, 9002, 9003) emitidas por la International Standards Organization (ISO) para el aseguramiento de la calidad constante.

Interfaces: Interfaz. Conexión e interacción entre hardware, software y el usuario. El diseño y construcción de interfaces constituye una parte principal del trabajo de los ingenieros, programadores y consultores. Los usuarios “conversan” con el software. El software “conversa” con el hardware y otro software. El hardware “conversa” con otro hardware. Todo este “diálogo” no es más que el uso de interfaces. Las interfaces deben diseñarse, desarrollarse, probarse y rediseñarse; y con cada encarnación nace una nueva especificación que puede convertirse en un estándar más, de hecho o regulado.

Interfaz: Circuito electrónico que gobierna la conexión entre dos dispositivos de hardware y los ayuda a intercambiar información de manera confiable. Es sinónimo de Puerto.

M

Método: Un enfoque sistemático para seguir una tarea. En un sentido amplio, modo ordenado de proceder durante el análisis y la resolución de situaciones. En tecnología orientada a objetos, se llama método al conjunto de reglas y procedimientos que rigen la manipulación y el acceso de los datos.

Modelo: Una abstracción de la realidad que presenta suficiente semejanza con el objeto del modelo que cuestiona sobre el objeto que puede ser examinado.

Modelo Entidad-Relación (E/R): Este modelo representa a la realidad a través de entidades que son objetos que existen y que se distinguen de otros por sus características, por ejemplo: un alumno se distingue de otro por sus características particulares como lo es el nombre, o el número de control asignado al entrar a una institución educativa, etc.

Modularidad: La independencia funcional de componentes de programa. Se divide el software en componentes identificables y tratables por separado, denominados módulos, que están integrados para satisfacer los requisitos del programa.

Monolitismo: m. Cohesión grande, solidez: el monolitismo de sus ideas garantiza una trayectoria estable.

N

Notación: Muchos métodos requieren crear descripciones abstractas, o modelos gráficos, del sistema en el análisis y/o diseño. Se construyen estos modelos usando una cierta forma de notación. La semántica de la notación proporciona el significado a los modelos. Las notaciones, para ser eficaces en el desarrollo de grandes sistemas, requieren un mecanismo para dividir los componentes en "pedazos más manejables".

O

Objeto: Los objetos son las cosas físicas y conceptuales que encontramos en el universo alrededor de nosotros. Hardware, software, documentos, seres humanos, los conceptos son todos los ejemplos de los objetos.

Ocultamiento de la información: Sugiere que los módulos se caractericen por decisiones de diseño que cada uno se oculte de los demás, es decir que la información (procedimientos y datos) contenida en cada módulo sea inaccesible a los demás módulos.

OMT (Object Modeling Technique).

OOD (Object Oriented Design).

OOP Programación Orientada a Objetos o POO (Object Oriented Programming).

OOSD (Object Oriented Structured Design): El método no es secuencial, el desarrollo del diseño es evolutivo, incremental e iterativo.

OOSE (Object Oriented Software Engineering): Tiene un enfoque llamado orientado a casos de uso. En este enfoque un modelo de casos de uso sirve como un modelo central del que todos los modelos se derivan.

Orientado a Función/Dato: Aquellos métodos en los cuales las funciones y/o los datos son tratados como entidades independientes.

P

Paradigma: Con el significado etimológico de *arquetipo*, un 'paradigma' es un modelo, y aplicado en este sentido a la lingüística, en especial a la gramática tradicional, se llama 'paradigma' a cada uno de los modelos, incluidas todas sus formas, de la flexión nominal (declinación) y verbal (conjugación). En la jerga informática un paradigma es una técnica, un modelo o un conjunto de herramientas.

Personal Software Process (PSP): Intenta desarrollar las actividades individuales y la disciplina a nivel personal para desarrollar software, y cuyo objetivo es mejorar la productividad de los desarrolladores mediante métodos de planeación, estimación y seguimiento de su trabajo.

Polimorfismo: Significa que una misma operación puede comportarse de modos distintos en distintas clases. La operación mover por ejemplo se puede comportar de modo distinto en las clases ventana y pieza de ajedrez.

Portabilidad: El esfuerzo necesario para transferir el programa de un entorno de sistema hardware y/o software a otro

Prototipos: Un sistema que funciona, desarrollado con la finalidad de probar ideas y suposiciones relacionadas con el nuevo sistema.

Procedimientos: Son los pasos que la organización define para la operación del sistema. Generalmente se encuentran establecidos en manuales.

Programación: Traducir a un lenguaje de programación las especificaciones del sistema. Codificar.

R

Relés: Son sistemas mediante los cuales se puede controlar una potencia mucho mayor con un consumo en potencia muy reducido. Es un interruptor operado magnéticamente. Este se activa o desactiva (dependiendo de la conexión).

S

SCRUM: Metodología ágil de desarrollo de software

Sinergia: El valor agregado que se crea al unir dos firmas separadas, y que permite obtener un retorno mayor a sus contribuciones individuales como entidades separadas. La ganancia total obtenida con la unión es mayor que la suma de sus partes.

SSADM: Análisis y diseño estructurado de sistemas (Structured Systems Analysis and Design).

Software: Conjunto de elementos lógicos de un sistema de información, como programas, reglas y procedimientos. Término genérico que se aplica a los componentes de un sistema informático que no son tangibles o físicos.

T

Técnica: Es un método que aplica herramientas y reglas específicas para completar una o más fases del ciclo de vida del desarrollo de sistemas.

Tecnología de Información (Information Technology IT): A menudo aparecen sólo sus iniciales: IT. Se trata de la tecnología necesaria para adquirir, almacenar, procesar y distribuir información por medios electrónicos: radio, televisión, teléfono, computadoras. El término abarca todas las formas de tecnología usadas para crear, almacenar, intercambiar y usar información en sus diversas formas (datos comerciales, conversaciones, imágenes estáticas, imágenes en movimiento, presentaciones multimedia y otras formas que incluyen lo que todavía no se concibió. IT incluye, en la misma palabra, telefonía y tecnología de la computación. Equivalencias: info-tecnología, tecnología de la información o tecnología

informática.

Técnicas de cuarta generación: En vez de requerir que quien desarrolla el software especifique los detalles procedimentales, un programa en un lenguaje no procedimental es la especificación del resultado deseado, en vez de la especificación de la acción requerida para conseguir el resultado. El software de soporte traduce la especificación del resultado en un programa maquina-ejecutable.

TSP (Team Software Process): Guía a los ingenieros entrenados en el uso de PSP, al equipo de desarrollo y a los gerentes en el desarrollo de grandes productos de software. Los ingenieros deben ser capaces de trabajar juntos de manera efectiva en un ambiente de equipo y saber como producir consistentemente productos de calidad en tiempo y costo.

U

UML: (Unified Modeling Lenguaje) El lenguaje para modelamiento unificado (UML), es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. Fue originalmente concebido por la Corporación Rational Software y tres de los más prominentes metodologistas en la industria de la tecnología y sistemas de información: Grady Booch, James Rumbaugh, e Ivar Jacobson. El lenguaje ha ganado un significativo soporte de la industria de varias organizaciones vía el consorcio de socios de UML y ha sido presentado al Object Management Group (OMG) y aprobado por éste como un estándar (noviembre 17 de 1997).

BIBLIOGRAFÍA

Libros

- Beck, K. *Extreme Programming Explained. Embrace Change*, Pearson Education, 1999. Traducido al español como: "Una explicación de la Programación Extrema. Aceptar el cambio", Ed. Addison Wesley, 2000.
- Booch Grady. *Análisis y Diseño Orientado a Objetos*, Ed. Addison Wesley / Díaz de Santos, Segunda edición 1998.
- Graham Ian. *Métodos Orientados a Objetos*, Ed. Addison –Wesley / Díaz de Santos, Segunda edición 1996.
- Kendall, KE & Kendall, JE. *Análisis y Diseño de Sistemas*, Prentice Hall Hispanoamericana, México 1994.
- Long, Larry E. *Introducción a las Computadoras y a los Sistemas de Información*, Ed. Prentice Hall, México 1998.
- Martín James, Odell James I. *Métodos Orientados a Objetos. Conceptos Fundamentales*, Ed. Prentice Hall, México 1997.
- McConnell Steve. *Desarrollo y Gestión de Proyectos Informáticos*, Ed. Mc. Graw Hill, España 1996.
- Monlau Pedro Felipe. *Diccionario Etimológico de la Lengua Española*, Ed. Compañía Impresora Argentina, S. A., Propiedad de Joaquín Gil, Librería "El Ateneo", 2ª. Edición 1944.
- Olle T. William, Hegelsten Jaques, Macdonald Ian G., Rolland Colette *Information Systems Methodologies*,

Piattini Mario G. *Elementos y Herramientas en el Desarrollo de Sistemas de Información*, Ed. Addison Wesley Iberoamericana, España 1995.

Pressman Roger S. *Ingeniería de Software*, Ed. Mc. Graw Hill, España 1998.

Rumbaugh, James. *Modelado y Diseño Orientado a Objetos, Informática y Computación*, Ed. Argos Vergara.

Senn James A. *Análisis y Diseño de Sistemas de Información*, Ed. Mc. Graw Hill, México 1998.

Whitten Jeffrey L., Bentley Lonnie D., Barlow Victor M. *Análisis y Diseño de Sistemas de Información*, Ed. Mc Graw Hill, Tercera edición, España 1998.

Yourdon Edward. *Análisis Estructurado Moderno*, Ed. Prentice Hall México 1997.

Diccionario de la Lengua Española. Real Academia Española. Vigésima segunda edición 2001. Ed. Espasa.

Informática y Computación. Informática Básica. Basic (1) primera parte. Ed. Argos Vergara, S. A. Barcelona 1986.

Diccionario de la Real Academia Española.

Compact Océano. *Diccionario Enciclopédico Color*. Grupo editorial Océano. Edición 1998.

Langensheidt Diccionario Básico Sinónimos y Antónimos. Grupo editorial Océano.

Tesis

Herrada Jiménez Jenny. *Documentación del Análisis y Diseño de Sistemas Orientados a Objetos con UML*. Licenciatura en Informática. UNAM, Facultad de Contaduría y Administración 2001.

Sánchez Hernández Olga. *Diseño Orientado a Objetos Fundamentos y Aplicaciones*. Ingeniero en computación. UNAM-Aragón 1995.

Santos Rosas Adriana. *La Metodología de Orientación a Objetos como nueva herramienta para el Análisis y Diseño de Software*, Ingeniero en computación. UNAM – Facultad de Ingeniería 1994.

Pineda Villalvazo Nino. *Un enfoque formal a la construcción de Software usando Eiffel*, Licenciatura en informática. UNAM – Facultad de Contaduría y Administración 1998.

Barrios Vázquez Marisela. *Metodología para el desarrollo de software: Diccionario de Términos Administrativos*, Licenciatura en informática. UNAM – Facultad de Contaduría y Administración 1997.

Jiménez Luna Silvia, Elvira Soriano Claudia. *UML y RUP como elementos de Sistemas Orientados a Objetos*, Licenciatura en informática. UNAM Facultad de Contaduría y Administración 2002.

Artículos de revistas

Novática. *Métodos de Desarrollo de Sistemas de Información*. Número 101, Enero-Febrero 1993.

<http://www.ati.es/novatica/1993/ene-feb/indice.html>

Páginas de Internet

Ingeniería de software de gestión.

“<http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/isoftware.htm>”

Ingeniería de Software. *Extracto de la tesis de maestría de [Zavala 2000].*

<http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html>

Ingeniería del software y sistemas de información.

“<http://issi.dsic.upv.es/>”

Taller sobre las *metodologías ágiles* de desarrollo de software.

“<http://issi.dsic.upv.es/tallerma>”

Sitio de la Asociación de Técnicos de Informática.

“<http://www.ati.es/>”

Página sobre metodologías de desarrollo de software.

<http://www.fdi.ucm.es/profesor/jpavon/doctorado/metodologias.pdf>

Sitio oficial sobre la metodología ágil de desarrollo denominada *programación*

extrema. <http://www.xprogramming.com/>

Página donde se puede colaborar para hacer conocer los principios de la

programación extrema. “<http://www.programacionextrema.org/>”.

Presenta información sobre fundamentos e historia de la *programación*
extrema.

“<http://www.geocities.com/chuidiang/metodologia/extrema.html>”

Desarrollo ágil de software. *Metodología ágil*.

“<http://www.agile-spain.com/downloads/learnagile.pdf>”

Metodologías ágiles en el desarrollo de software. *Metodología ágil*.
“<http://www.willydev.net/descargas/prev/TodoAgil.Pdf>”

Metodologías de desarrollo de software. *Metodología ágil*.
“<http://www.acm.org/crossroads/espanol/xrds7-4/intro74.html>”

Metodologías de desarrollo de software. *Metodología ágil*.
“<http://www.ajlopez.net/ajlopez/ArticuloVe.php?Id=538>”

PROSOFT 2003
<http://www.software.net.mx>

Sitio oficial de la compañía de R. S. Pressman y asociados.
“<http://www.rspa.com/>”.

IDS. Empresa consultora líder en el desarrollo de software que hace uso de metodologías, estándares y herramientas aunado a la capacitación constante en las tendencias tecnológicas.
“http://www.ids.com.mx/Desarrollo/desarrollo_software.asp”

Sitio oficial de IBM en donde se describen las principales plataformas de desarrollo de software y además se muestran nuevas herramientas y programas para el desarrollo de software con calidad.
“<http://www-306.ibm.com/software/rational/>”