



Ingeniería de Software

Repaso para CENEVAL

- Ing. Rafael Salazar Chávez
- Ing. Gustavo Cervantes Orne
- Ing. Jakeline Marcos Abed

Repaso CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Temas

- Ciclo de vida de sistemas
 - Las fases del desarrollo de software
 - Ciclos de vida de desarrollo de sistemas
- Metodologías para el desarrollo de sistemas
 - Enfoque funcional
 - Enfoque de datos
 - Enfoque orientado a objetos
 - Herramientas CASE
- Administración del desarrollo y la calidad del sistema
 - Administración de proyectos
 - Aseguramiento de la calidad
- Industria del software

Bibliografía Recomendada

- Pressman, Roger. Ingeniería del software, Un enfoque práctico (5a ed.), McGraw-Hill, México, 2002.
 - http://www.mhhe.com/engcs/compsci/pressman/student_index.mhtml
- Sommerville, Ian. *Software engineering* (7a. ed.), Addison-Wesley, 2004.
- Marchewka, Jack T. Information Technology Project Management: Providing measurable Organizational Value. 2006. Wiley. ISBN: 0-471-71539-5
- Jalote, Pankaj. Software Project Management in Practice. Addison-Wesley Professional. 2002. ISBN-10: 0201737213. ISBN-13: 978-0201737219
- Software Project Survival Guide: How to Be Sure Your First Important Project Isn't Your Last. Redmond, Wa.: Microsoft Press, 288 pages, 1997.
- www.swebok.org

Tema: Metodologías para el desarrollo de sistemas

- Enfoque funcional (Yourdon)
 - Análisis estructurado
 - Diseño estructurado
- Enfoque de datos
 - Modelo entidad-relación
- Enfoque orientado a objetos
 - Conceptos generales
 - Análisis orientado a objetos
 - Diseño orientado a objetos
- Herramientas CASE

Tema: Administración del dlio y calidad del sistema

- Administración de proyectos
 - Estimación de tiempo y costo de cada actividad
 - Planificación temporal de proyectos
- Aseguramiento de la calidad
 - Concepto de calidad
 - Estándares
 - Especificaciones
 - Factores de calidad
 - Métricas de calidad
 - Revisiones técnicas

Tema: Industria del software

- Proceso de desarrollo
- Pruebas betas
- Control de versiones
- Soporte técnico

Introducción

¿Qué es Software?

- Las **instrucciones** (programas de computadora) que al ser ejecutados proveen una funcionalidad y desempeño esperado, las **estructuras de datos** que permiten que los programas manipulen la información de manera adecuada y la **documentación** que describe la operación y el uso de los programas

¿Qué es Software?

- Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

IEEE-729

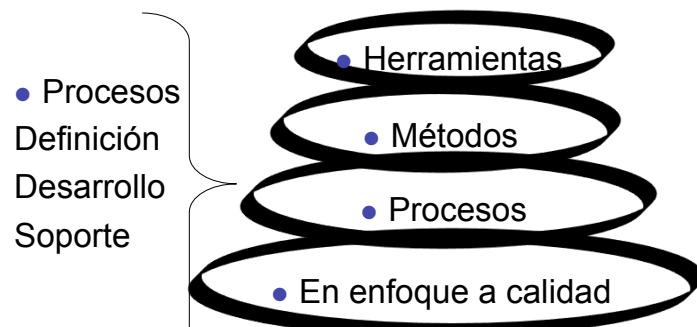
Tipos de Software

- Software **del Sistema**: eje. S.O., compiladores, etc.
- Software **de Negocios**: eje. ERP, POS,etc
- Software **ingenieril y científico**: Eje. CAD, simulación, etc
- Software **embedido**: residente en memoria (ROM) y se utiliza principalmente para funciones de control de equipos
- Software para **cómputo personal**: eje. Hojas de cálculo, procesadores de textos, etc
- Software **basado en Web**: eje. Las páginas de web que son accesibles vía un navegador

Ingeniería de software

- Def. IEEE. La aplicación **sistemática, disciplinada y medible**, orientada al **desarrollo, operación y mantenimiento del software**, esto es, la aplicación de métodos y prácticas de ingeniería al software.
- Def. El establecer y utilizar principios de ingeniería para obtener software que es confiable y que funciona eficientemente en equipos de cómputo y dentro parámetros económicos sustentables

Capas de la Ingeniería de Software



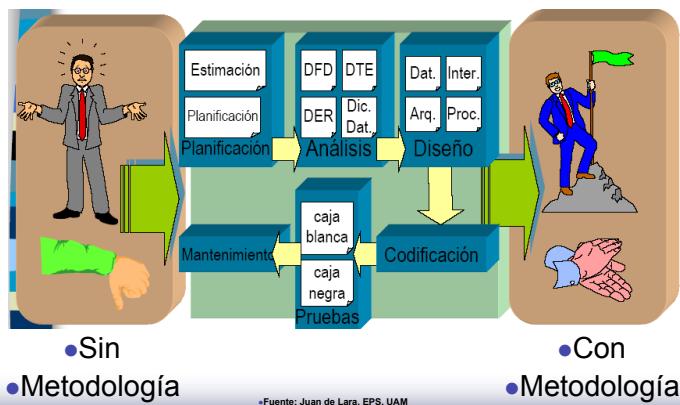
Ingeniería de software utiliza:

- **Métodos:** proveen las **diferentes reglas y fases** para el desarrollo de SW:
 - Project Planning and Estimation
 - Requirement Analysis
 - Design
 - Algorithm Development
 - Coding
 - Testing
 - Deployment
- **Herramientas apoyo automatizado** para ejecutar los métodos:
 - MS Project
 - CASE Tools
 - Coding Tools
 - Testing Tools
- **Procesos** define la secuencia en la cual los métodos serán ejecutados para producir las **salidas esperadas**, con la **calidad esperada**.

Actividades Complementarias de la ing. de software: Producción de Sistemas

- **Seguimiento y control** del proyecto
- **Revisión** técnicas formales
- Aseguramiento de **calidad** del software
- Administración de la **configuración**
- Preparación y producción de la **documentación**
- Gestión de la **reusabilidad** : componentes
- **Métricas**
- Gestión de **riesgos**

Metodologías



Ingeniería de Software

- Desde 1968 hasta la fecha diferentes organismos (SEI, IEEE, ISO) han enfocado sus esfuerzos en:
 - **Identificar** los **factores** clave que determinan la **calidad** del software.
 - **Identificar** los **procesos** necesarios para producir y mantener software.
 - **Definir** la **base de conocimiento** necesaria para la producción y mantenimiento de software.

- El resultado ha sido la **necesidad de profesionalizar el desarrollo, mantenimiento y operación de los sistemas de software**, introduciendo métodos y formas de trabajo sistemáticos, disciplinados y cuantificables.

Ingeniería de Software: Principales organizaciones de estandarización.

▪**ISO: International Organization for Standardization**

▪**IEEE- Computer Society: Institute of Electrical and Electronics Engineers**

▪**SEI: Software Engineering Institute**



Ingeniería de Software: Principales organizaciones de estandarización.

▪ ISO

- Organización Internacional para la Estandarización. Fundada en 1947
- Son miembros 87 países.
- En 1987 la Organización Internacional para la Estandarización (ISO) y la Comisión Internacional Electrotécnica (IEC), establecieron un Comité Internacional (JTC1) para las Tecnologías de la Información. La misión del JTC1 es la "estandarización en el campo de los sistemas de tecnologías de la información, incluyendo microprocesadores y equipos.
- Los estándares o instrucciones técnicas más importantes para la Ingeniería del Software:
 - ISO/IEC 12207: Procesos del ciclo de vida del software
 - ISO/IEC TR 15504 Marco para métodos de evaluación, no es un método o modelo en sí.

Ingeniería de Software: Principales organizaciones de estandarización.

▪ SEI



- Instituto de Ingeniería del software.
- (SEI <http://www.sei.cmu.edu/>).
- Integrado en la Universidad Carnegie Mellon.
- La aportación más significativa: Los modelos de madurez de las capacidades: PSP/TSP, CMM y CMMI; que son un **marco de referencia para mejora de procesos**, y **criterio de evaluación para determinar la madurez**, y por tanto determinar la fiabilidad de resultados previsibles de una organización de software.



Ingeniería de Software: Principales organizaciones de estandarización.

▪ IEEE Computer Society

- IEEE Es el Instituto de Ingenieros en electricidad y electrónica (Institute of Electrical and Electronics Engineers).
- Su misión es preservar, investigar y promover la información de las tecnologías eléctricas y electrónicas.
- Surgió en 1963 con la fusión del AIEE (Instituto Americano de Ingenieros Eléctricos) y el Instituto de Ingenieros de Radio (IRE).
- La IEEE Computer Society (www.computer.org) es una sociedad integrada en IEEE, formada en la actualidad por más de 100.000 miembros en todo el mundo.
- Su finalidad es avanzar en la teoría, práctica y aplicación de las tecnologías de la información. Realiza conferencias, publicaciones, cursos de formación, y desarrolla estándares.

Ingeniería de Software: Principales organizaciones de estandarización.



■ IEEE Computer Society

● Estándares para la Ingeniería del Software

- IEEE ha desarrollado estándares para todas las áreas de Ingeniería del Software. Algunos de ellos, correspondientes a las principales áreas específicas de la Ingeniería del Software son:

- IEEE Std. 830 Prácticas recomendadas para las especificaciones de software.
- IEEE Std. 1362 Guía para la especificación del documento de requisitos "ConOps"
- IEEE Std. 1058 Standard for Software Project Management Plans
- IEEE Std. 1063 Estándar para la documentación de usuario de software.
- IEEE Std. 1012 Estándar para la verificación y validación de software.
- IEEE Std. 1219 Estándar para el mantenimiento del software

21

• Fuente: Juan Palacio

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM



Ingeniería de Software: SWEBOk.

- El proyecto SWEBOk (Software Engineering Body of Knowledge) comenzó sus actividades de manera efectiva dentro del SWECC¹ en 1997 (aunque el comité SWECC se creó en 1993).
- En el proyecto también están representados:
- los dos principales organizaciones de estandarización en Ingeniería del Software: IEEE e ISO/IEC JTC1/SC/.
- Los autores de las tres principales obras de Ingeniería del Software: Steve McConnell, Roger Pressman e Ian Sommerville.
- Empresas y organizaciones como: [Rational](#), [SAP](#), [Boeing](#), [Construx](#), [MITRE](#), [Raytheon](#),
- En 2001 el proyecto publicó ya una definición consensuada del cuerpo de conocimiento aceptado en la ingeniería del software (<http://www.swebok.org>).
- **El cuerpo de conocimiento identificado por el proyecto SWEBOk se ha configurado como el estudio más relevante y como la referencia de más autoridad en toda la comunidad informática para la acotación y descripción de los conocimientos que configuran la Ingeniería del software.**



1 Software Engineering Coordinating Committee. Comisión creada por IEEE Computer Society y ACM (Association for Computer Machinery) para definir el cuerpo de conocimiento de la ingeniería del software.

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Ingeniería de Software: Principales organizaciones de estandarización.

La Ingeniería del Software es una ingeniería muy joven que necesitaba:

1. **Definirse a sí misma:** ¿Cuáles son las áreas de conocimiento que la comprenden?
[SWEBOk: Software Engineering Body of knowledge](#)
2. **Definir los procesos que intervienen** en el desarrollo, mantenimiento y operación del software
[ISO/IEC 12207: Procesos del ciclo de vida del software](#)
3. **De las mejores prácticas, extraer modelos** de cómo ejecutar esos procesos para evitar los problemas de la “crisis del software”
[CMM / CMMI](#)
[ISO/IEC 15504: Marco para métodos de evaluación, no es un método o modelo en sí.](#)
4. **Definir estándares menores** para dibujar criterios unificadores en requisitos, pruebas, gestión de la configuración, etc.
[IEEE 830 - IEEE 1362 - ISO/IEC 14764 ...](#)

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM



Ingeniería de Software: SWEBOk.

- **SWEBOk:** Necesidad de establecer cuál es el cuerpo de conocimiento que deben conocer los ingenieros del software, y en su desarrollo ha agrupado este conocimiento en **10 áreas**:

1. Requerimientos
2. Diseño
3. Construcción
4. Pruebas
5. Mantenimiento
6. Gestión de la configuración
7. Gestión de Ing. Sw.
8. Procesos de Ing. Sw.
9. Herramientas y métodos
10. Calidad

Es importante resaltar que estas áreas no incluyen aspectos clave de las tecnologías de la información, tales como lenguajes específicos de programación, bases de datos, o tecnología de redes y comunicaciones.



Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

24

Ingeniería de Software: Principales organizaciones de estandarización.

ISO 12207 Propósito:

Establecer un estándar para proporcionar un marco y un lenguaje común en la disciplina del software.

No se trata de un estándar de certificación, tipo ISO 9000, sino de un estándar para la normalización.

Establece un marco común para el ciclo de vida del software en:

- Adquisición, suministro, desarrollo, operación y mantenimiento del software
- Gestionar, controlar y mejorar el marco como base de referencia para el trabajo e intercambio entre organizaciones de software

Ciclo de vida del software

Periodo de tiempo que comienza al concebir la idea de un nuevo sistema de software, y termina cuando este se retira y deja de funcionar.



TECNOLÓGICO
DE MONTERREY.

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Ciclo de vida del proyecto

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Ingeniería de Software: Principales organizaciones de estandarización.

ISO 12207 Propósito:

El estándar **no** prescribe:

- Que deba emplearse ningún tipo de documentación específica.
- Que deba emplearse un tipo específico de ciclo de desarrollo.
- Métodos concretos para el desarrollo, mantenimiento u operación del software.

Define el QUÉ, no el CÓMO. Dice cuáles son los procesos, actividades y tareas implicados en el desarrollo, mantenimiento y operación de los sistemas de software, asentando un marco estándar de referencia internacional, pero no se ocupa ni prescribe técnicas específicas.

El estándar sirve de referencia desde dos perspectivas diferentes:

- Para la adquisición de sistemas y servicios de software.
- Para el suministro, desarrollo, mantenimiento y operación de productos de software.

El estándar no cubre el desarrollo de productos de software para distribución comercial masiva (productos 'en caja').



Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

El ciclo de vida del proyecto

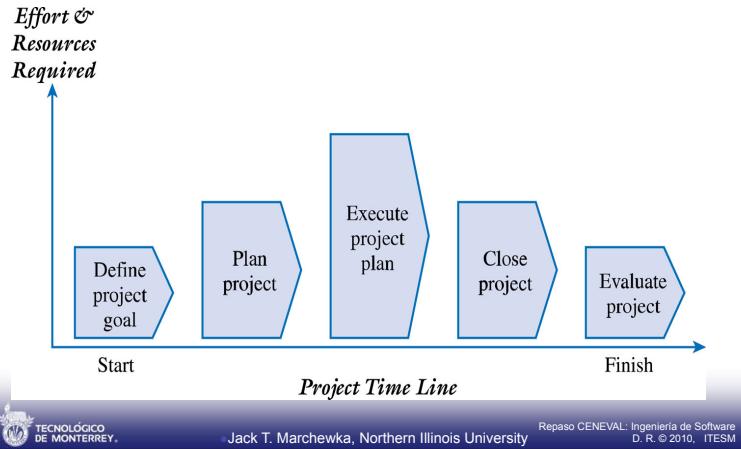
• Project Life Cycle (PLC)

- Una **colección de etapas lógicas o fases** que mapea la vida de un proyecto desde su inicio hasta su fin, para definir, construir y entregar el producto del proyecto – es decir, el Sistema de Información.
- Los proyectos se dividen en fases para hacerlo más manejable y reducir los riesgos.
 - *Phase exits, stage gates, or kill points* son puntos de decisión al final de cada fase para evaluar el desempeño, corregir problemas o cancelar el proyecto.
 - *Fast tracking* es el traslape de las fases para disminuir los tiempos del proyecto.
 - Puede ser riesgoso !



Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

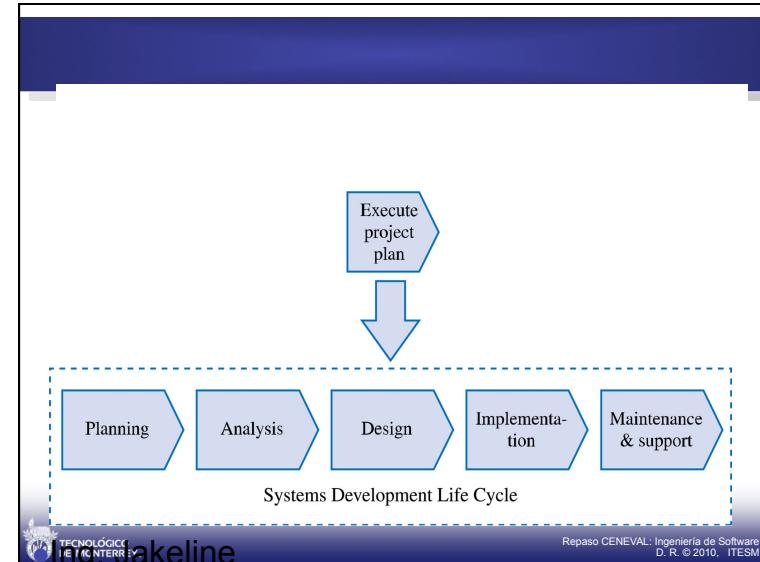
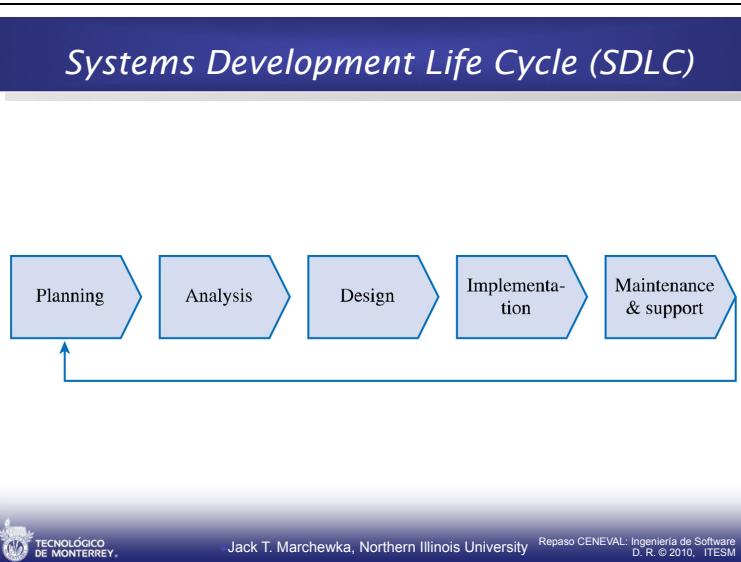
Generic Project Life Cycle



Systems Development Life Cycle (SDLC)

- Representa las fases secuenciales que un Sistema de Información sigue a lo largo de su vida.
- Fases:
 - Planeación
 - Análisis
 - Diseño
 - Implementación
 - Mantenimiento y soporte

Systems Development Life Cycle (SDLC)



Relación entre PLC y SDLC

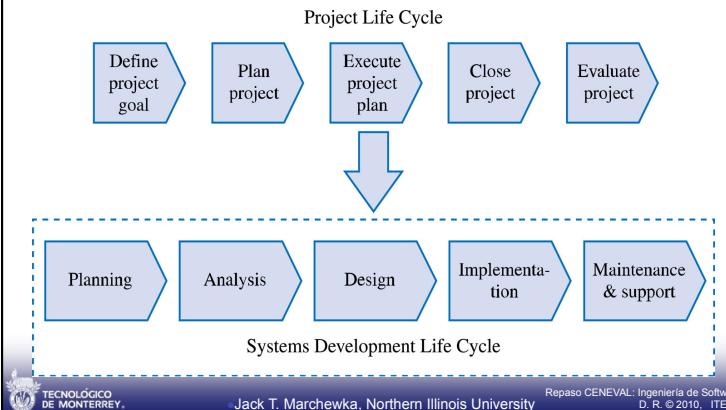
- El ciclo de vida de desarrollo de sistemas (SDLC) es parte del ciclo de vida del proyecto (PLC).
 - El **PLC** se enfoca en las fases de administración de proyectos, procesos, herramientas y técnicas para administrar efectivamente el proyecto.
 - El **SDLC** se enfoca en las fases de ingeniería de SW, procesos, herramientas y técnicas para construir o implementar una solución de IT.



TECNOLÓGICO
DE MONTERREY.

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Relación entre PLC y SDLC



TECNOLÓGICO
DE MONTERREY.

• Jack T. Marchewka, Northern Illinois University

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

SDLC systems development life cycle

- Los productos tienen ciclos de vida
- **systems development life cycle (SDLC)** es un marco para describir las fases involucradas para desarrollar sistemas de información.
- Pueden ser:
 - **Predictivas**: El *alcance* del proyecto puede ser planeado y el *costo* y el *calendario* se pueden predecir.
 - **Adaptativas**: Proyectos creados de acuerdo a su misión y basados en componentes, muy específicos en tiempo, y para cumplir ciertos objetivos de fechas.



TECNOLÓGICO
DE MONTERREY.

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Predictive Life Cycle Models

- **Waterfall model**: tiene etapas muy bien definidas.
- **Spiral model**: Desarrollado en forma iterativa
- **Incremental build model**: desarrollo progresivo para SW de operación
- **Prototyping model**: Utilizado para clarificar requerimientos.
- **Rapid Application Development (RAD) model**: Usado para producir sistemas rápidamente, sin sacrificar calidad.



TECNOLÓGICO
DE MONTERREY.

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Adaptive Life Cycle Models

- **Extreme programming (XP):** Desarrollo en parejas, probando su propio código. Los equipos de XP incluyen desarrolladores, administradores y usuarios.
- Característica de muchos proyectos actuales que involucran: velocidad, incertidumbre, req. cambiantes y riesgo alto.
- XPM se basa en que los proyectos casi siempre son caóticos e impredecibles.
- XPM se enfoca en flexibilidad, adaptabilidad e innovación.
- **Scrum:** desarrollo iterativo en el que las repeticiones son llamadas *sprints*, y normalmente dura 30 días. Los equipos se reúnen cada día, en una pequeña junta llamada *scrum*, para decidir qué se va a hacer ese día. Se adapta mejor a la tecnología orientada a objetos y los proyectos requieren un fuerte liderazgo para coordinar la labor. Es una forma de auto-gestión de los equipos de programadores. *Un grupo de programadores deciden cómo hacer sus tareas y cuánto van a tardar en ello. Scrum ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro*



TECNOLÓGICO
DE MONTERREY.

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Fases de Desarrollo de SW

- **Análisis preliminar** (enunciado general del problema, propósito del sistema, requerimientos, principales entradas y salidas, diagrama de bloques del sistema, estimación de costo-beneficio y plan preliminar de desarrollo)
- **Análisis detallado** (técnicas de recolección de información, determinación de los requerimientos detallados, formalización de los requerimientos, especificación de alcances y limitaciones del sistema)
- **Diseño preliminar** (definición de restricciones, diseño de la arquitectura del sistema, definición de interfaces entre módulos)
- **Diseño detallado** (descomposición de los módulos en componentes elementales, especificación de los componentes elementales, diseño detallado de entradas y salidas, diseño detallado de la base de datos)
- **Construcción** (programación estructurada u orientada a objetos, pruebas de componentes)
- **Pruebas** (elaboración del plan de pruebas, preparación de casos, establecimiento de criterios de revisión, elaboración de las pruebas, evaluación de resultados)
- **Instalación** (manual de usuario, ayudas interactivas, capacitación, corridas de prueba)
- **Mantenimiento** (correctivo, adaptativo, perfectivo, pruebas de regresión, técnicas de mantenimiento)



TECNOLÓGICO
DE MONTERREY.

39

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Ciclo de vida de sistemas

- **Las fases del desarrollo de software**
 - Análisis preliminar
 - Análisis detallado
 - Diseño preliminar
 - Diseño detallado
 - Construcción
 - Pruebas
 - Instalación
 - Mantenimiento
- **Ciclos de vida de desarrollo de sistemas**
 - En cascada
 - Espiral



38

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Modelos de ciclo de vida de SW

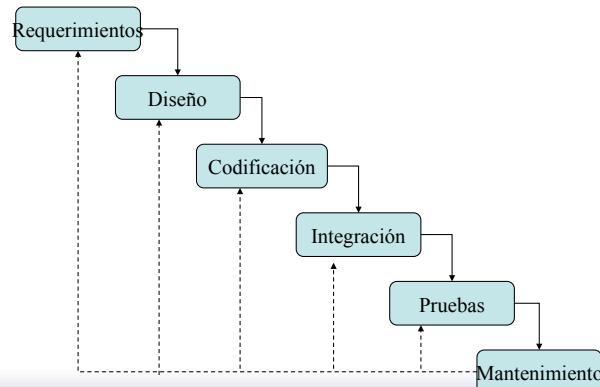


Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

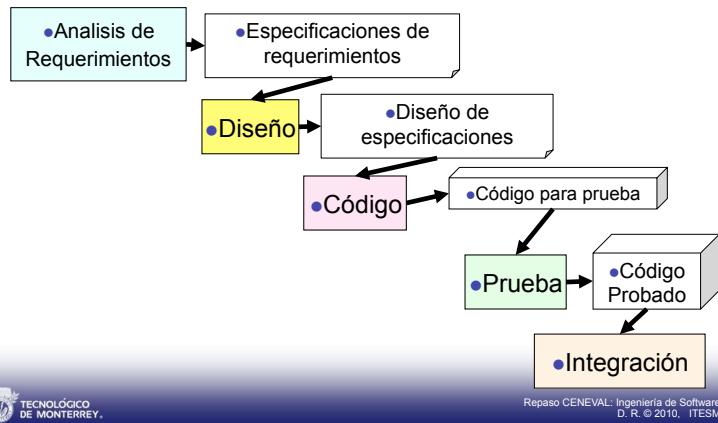
Modelos del Ciclo de Vida

- Secuenciales o lineales: cascada
- Prototipos: Funcionales, navegacionales
- Rápidos o Acelerados: RAD
- Evolutivos: Incremental, Espiral, Desarrollo concurrente
- Basado en componentes: OO
- Formales: métodos matemáticos

Modelo de Cascada



Modelo de Cascada del Ciclo de Vida del SW



Desventajas del modelo cascada:

Asume que:

- Los requerimientos están claramente comprendidos.
- Las fases ocurren secuencialmente

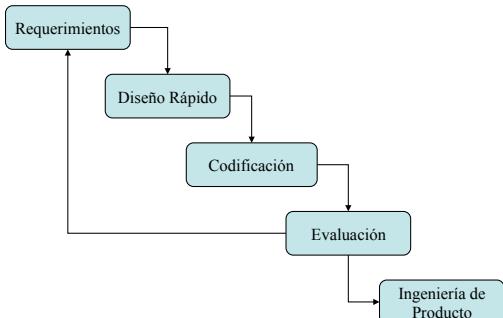
Ventajas:

- Modelo simple y sistemático
- Con un enfoque disciplinado

Desventajas:

- No acepta cambios en los reqs.
- No identifica los riesgos de manera temprana.

Prototipos



45

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Modelo Prototipos

Ventajas:

- Menos riesgos técnicos
- Considera nuevos reqs.
- Puedes visualizar el producto de manera temprana

Desventajas:

- Caro
- Consumo mucho tiempo

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Modelo de Spiral

- Originado por Barry Boehm en 1970's
- Idea básica: el SW es desarrollado en **una serie de prototipos sucesivos más capaces**, cada uno de los cuales es diseñado en etapas anteriores enfocándose en las áreas de alto riesgo
- Objetivo: el mismo que **waterfall**: desarrollar, producir y apoyar un producto de SW
- Diferencia: serie de etapas pequeñas que evolucionan, en vez de una sola secuencia.
- Utilizado para proyectos **grandes, muy complejos**.

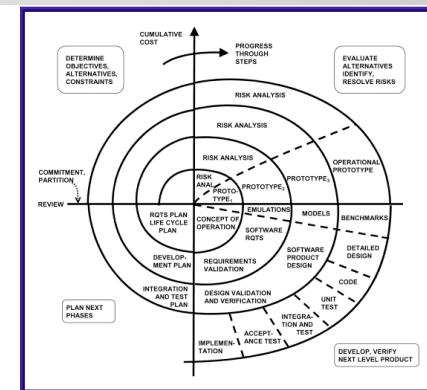
Boehm, Barry, "A Spiral Model of Software Engineering," IEEE Transactions on Software Engineering,

1970



Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

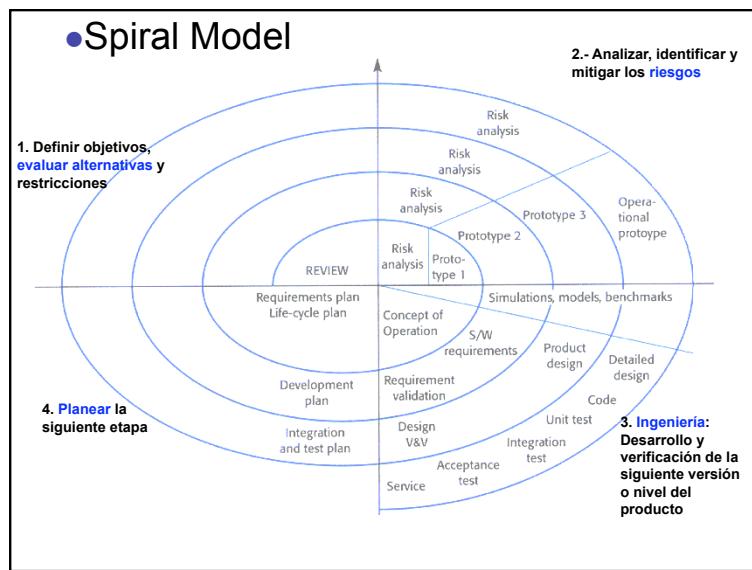
Espiral



Fuente: Barry W. Boehm,

48

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM



Modelo de espiral

Ventajas:

- Se puede ajustar a los cambios en requerimientos
- Facilita la administración de riesgos
- Desarrollo incremental y evolutivo
- A medida que el costo del proyecto se incrementa, el riesgo decrece.

Desventajas:

- Admite poco entendimiento del problema
- Es difícil decir dónde estás exactamente y cuánto te falta de hacer
- NO hay criterios de salida o puntos de control en el tiempo
- Es difícil administrar y puede ser muy costoso

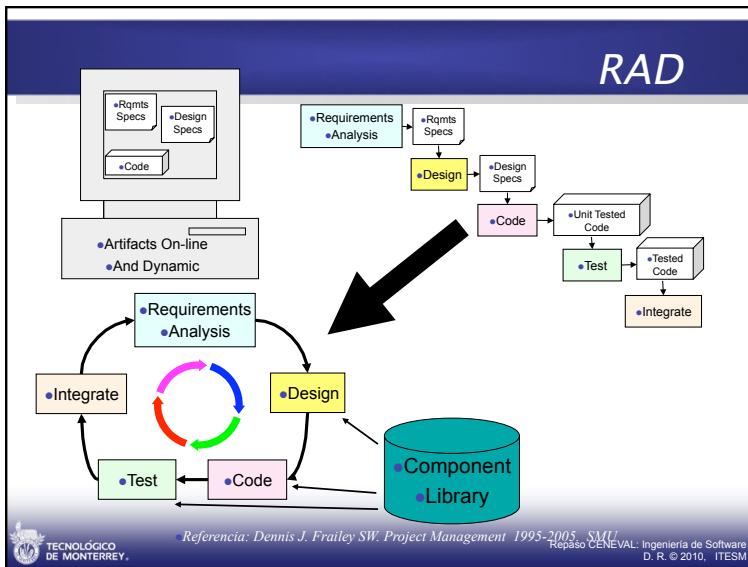
Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM



Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM



Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM



Ventajas y Desventajas RAD Model

Pros

- Puedes mostrar muy pronto un producto trabajando. Cada iteración termina con un producto usable
- Los usuarios están involucrados a lo largo de todo el ciclo de vida.
- Actualización continua de la documentación, lo que ayuda a futuras mejoras
- Adapta los cambios en Requerimientos
- Enfrenta de manera efectiva uno de los mayores riesgos en este tipo de SW que es fallas en los requerimientos del mercado!

Cons

- No hay mucho seguimiento de estándares
- Es difícil saber dónde vas
- Puede ser caótico si no se administra efectivamente.
- Verificación y pruebas de calidad exhaustivas, No son parte del proceso

Extreme Programming (XP)

- Una disciplina “ligera” de desarrollo de software derivada de *object oriented programming* y basada en los siguientes principios:
 - Simplicidad
 - Comunicación entre los desarrolladores
 - Retroalimentación
 - Mucha motivación !
- Originada con el proyecto “C3” de Chrysler Corporation in 1997

Definición simple de XP

- Forma altamente iterativa de RAD donde el enfoque es minimalista
 - Diseño evolucionario en vez de diseño planeado
 - Tipicamente ciclos de 3 semanas, aún para proyectos muy grandes
 - No planean NADA, porque lo más probable es que estés mal !
 - Solo hace lo que es necesario en este momento – agregas funcionalidad después

Supuestos de XP:

- Equipos pequeños
- Clientes cambiarán su forma de pensar
- Un producto perfecto no puede ser producido, entonces saca algo y después lo mejoras
- Mucho contacto entre los programadores y el cliente.

Prácticas clave para XP:

- Refactoring – rediseño constante para responder a los cambios
- Primero prueba y después codifica
- Pair programming – 2 personas inspeccionando el trabajo de los demás
- Integración continua

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Desventajas XP

- La simplicidad no siempre se puede llevar a cabo
- Es muy fácil perder la disciplina
- No se ajusta bien a proyectos grandes
- Todo se basa en que los desarrolladores entiendan bien el problema – No hay especificaciones !

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Desarrollo Incremental

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Modelos del Ciclo de Vida

• Incremental

El modelo incremental mitiga la rigidez del modelo en cascada, **descomponiendo el desarrollo de un sistema en partes**; para cada una de las cuales se aplica un ciclo de desarrollo:

- Las ventajas que ofrece son:
 - El usuario dispone de **pequeños subsistemas** operativos que ayudan a perfilar mejor las necesidades reales del sistema en su conjunto.
 - El modelo **produce entregas parciales** en períodos cortos de tiempo, comparados con el tiempo necesario para la construcción del sistema en su conjunto, y permite la incorporación de nuevos requisitos que pueden no estar disponibles o no ser conocidos al iniciar el desarrollo.

Referencia: Juan Palacio
60

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Modelos del Ciclo de Vida

•Incremental

•Aunque en la representación gráfica de la figura anterior, los desarrollos de **cada subsistema se solapan en el tiempo**, en su aplicación real, el segundo y siguientes subsistemas pueden comenzar una vez concluido el anterior.

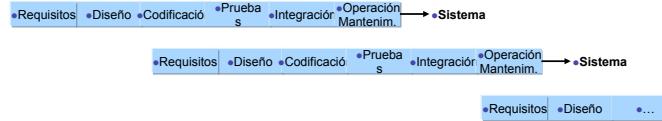
Resulta apropiado:

- Desarrollo de sistemas en los que **el cliente necesita disponer de parte de la funcionalidad antes** de lo que costaría desarrollar el sistema completo.
- Desarrollo de sistemas en los que por razones del contexto interesa **realizar la obtención de los requisitos de forma escalonada** a través de subsistemas.



Modelos del Ciclo de Vida

•Evolutivo



- Este modelo está compuesto por varios ciclos de desarrollo. Cada uno de ellos produce un sistema completo, con el que se operará en el entorno de operación.
- La información acumulada en el desarrollo de cada sistema, y durante su fase de operación sirve para **mejorar o ampliar los requisitos y el diseño del siguiente**.
- En realidad es un ciclo de vida común a todos los sistemas desarrollados que se mejoran a través de versiones sucesivas.



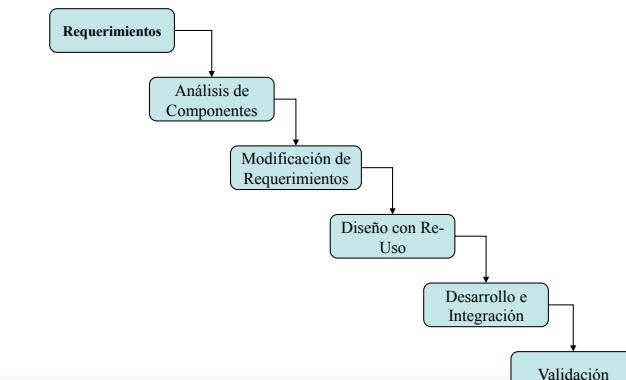
Modelos del Ciclo de Vida

•Evolutivo

- Las circunstancias en las que este modelo puede resultar apropiado son
 - Desconocimiento inicial de todas las necesidades operativas que serán precisas, generalmente por tratarse del desarrollo de un sistema que operará en **un entorno nuevo sin experiencia previa**.
 - Necesidad de que el sistema entre en operación en **tiempos inferiores** a los que serían necesarios para diseñarlo y elaborarlo de forma exhaustiva.
 - Necesidad de desarrollar sistemas en **entornos cambiantes** (sujetos a normas legislativas, mejora continua del producto para hacer frente a desarrollos de la competencia, etc.).
- Aunque en su concepción inicial contempla desarrollos internos en cascada, también podría plantearse, por ejemplo, un ciclo de vida evolutivo con desarrollos internos en espiral.



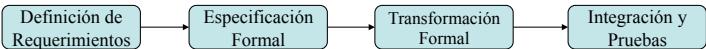
Desarrollo Orientado al Reuso



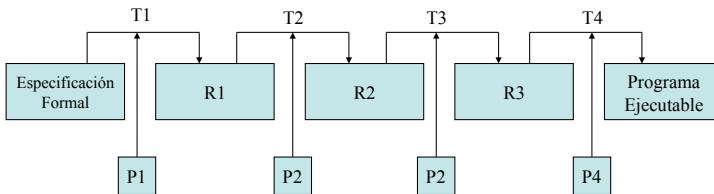
Desarrollo Formal de Sistemas

- Se basa en la transformación de una especificación matemática a través de representaciones de un programa ejecutable
- Las transformaciones deben preservar “la exactitud” y conformidad hacia adelante, mostrando el cumplimiento de la especificación.

Desarrollo formal de sistemas



Transformaciones de Programas



Análisis y Diseño

Ingeniería de Requerimientos

- ¿Qué es un requerimiento?
- ¿Cuántos tipos de requerimientos hay?
- ¿Cuál es el proceso para administrar los requerimientos?

Ingeniería de Requerimientos

- ¿Qué es un requerimiento?
- Una condición o capacidad requerida por el usuario para resolver un problema o alcanzar un objetivo.
- Una condición o capacidad que debe poseer un sistema o un componente, para satisfacer un contrato, estándar, especificación, o algún otro documento formal. El conjunto de todos los requerimientos formará las bases para los desarrollos subsecuentes del sistema o del componente.

Técnicas de Adquisición

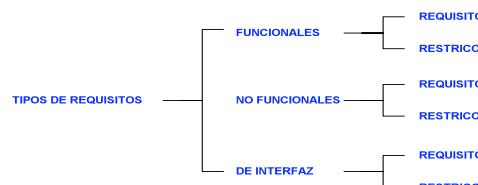
- **Entrevistas:** Reuniones, cuestionarios, reuniones de grupo, entrevista, lluvia de ideas
- **Juntas dirigidas**
- **Prototipos:** Prototipos rápidos, prototipos evolutivos
- **Observación:** Introspección, análisis de protocolo, documentación, otros sistemas
- **Escenarios:** Casos de uso, tarjetas CRC, diagramas de flujo, escenarios
- **Lluvia de ideas**

Proceso

- **Adquisición (elicitation)**
 - Identificar los requisitos que se obtienen de los usuarios y clientes.
- **Análisis**
 - Razonar sobre los requisitos adquiridos, combinar requisitos relacionados, establecer prioridades entre ellos, determinar su viabilidad, etc.
- **Especificación**
 - Registrar los requisitos de alguna forma, incluyendo lenguaje natural, lenguajes formales, modelos, prototipos, maquetas, etc.
- **Validación**
 - Examinar inconsistencias entre requisitos, determinar la corrección, ambigüedad, etc. Establecer criterios para asegurar que el software reúna los requisitos cuando se haya producido. El cliente, usuario y desarrollador se deben poner de acuerdo.

Requisitos del software

Clasificación de los requisitos



■ Requisitos funcionales

- Definen el comportamiento del sistema.
- Describen las tareas que el sistema debe realizar.
- Al definir un requisito funcional es importante mantener el equilibrio entre la excesiva generalidad, insuficiencia de detalle o ambigüedad, y el exceso de detalle con precisiones o descripciones innecesarias o redundantes.

Requisitos del software

Clasificación de los requisitos

- **Requisitos no funcionales**
 - Definen aspectos, que sin ser funcionalidades, (tareas que el sistema debe realizar) resultan deseables desde el punto de vista del usuario. Generalmente comprenden atributos de calidad:
 - Tiempos de respuesta.
 - Características de usabilidad.
 - Facilidad de mantenimiento.
 - etc.

■ Requisitos de interfaz

- Definen las interacciones del sistema con su entorno:
 - Usuarios
 - Otros sistemas

Referencia: Juan Palacio
73

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

 TECNOLÓGICO DE MONTERREY.

Requerimientos ideales

- **Completo** (Cada requerimiento es conocido y asignado)
- **Suficiente** (Cada característica requerida es especificada)
- **Consistente** (Ningún requerimiento se contradice con algún otro)
- **Sin ambigüedad** (Una sola posible interpretación)
- **Probable** (Se puede probar cuando se cumplen los req.)

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

 TECNOLÓGICO DE MONTERREY.

Técnicas de Análisis

- Funcionales (análisis estructurado)
- Orientada a Datos
- Orientadas a Objetos

75

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

 TECNOLÓGICO DE MONTERREY.

Análisis Estructurado

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

 TECNOLÓGICO DE MONTERREY.

Elementos del Análisis Estructurado

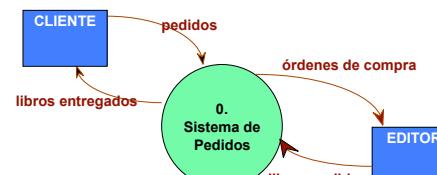
- Diagrama de Flujo de Datos
- Diccionario de Datos
- Descripción de procesos
 - Lenguaje estructurado
 - Pre y post-condiciones
 - Tablas de decisión
 - Árboles de decisión
- Diagrama de Entidad Relación
- Diagrama de Transición de Estados

DFD

Elemento	Campos típicos en una Herramienta CASE	Símbolo
Cada proceso tiene:	<ul style="list-style-type: none"> Etiqueta (nombre) Tipo (proceso) Descripción (qué es) Número de proceso Descripción de proceso (lenguaje estructurado). Notas 	Proceso <número> <Nombre>
Cada flujo de datos tiene:	<ul style="list-style-type: none"> Etiqueta (nombre). Un número. Un nombre o frase verbal Una descripción Uno o más flujos de datos de salida. Normalmente uno o más flujos de entrada. 	Flujo Datos <Nombre>
Cada Almacén de datos tiene:	<ul style="list-style-type: none"> Etiqueta (nombre) Un número Un nombre (un sustantivo). Una o más conexiones a un proceso Una o más conexiones a un flujo de datos. Normalmente uno o más flujos de salida 	Almacén Datos D<numero> Nombre
Cada entidad externa tiene:	<ul style="list-style-type: none"> Etiqueta (nombre) Un nombre (sustantivo) Una descripción 	Entidad <Nombre>

Fuente: Juan de Lara, EPS, UAM

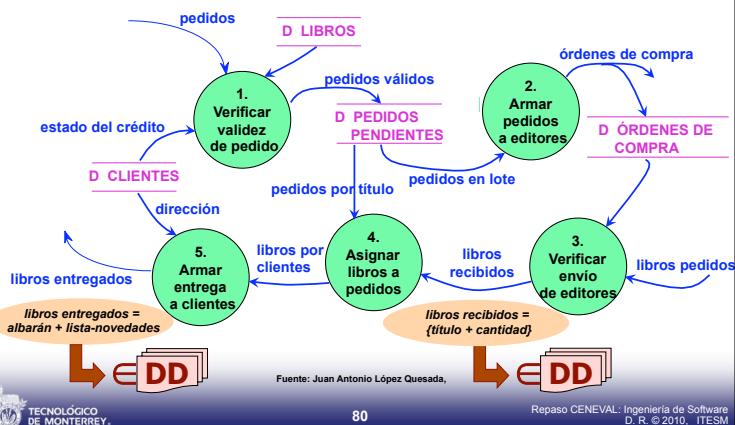
DFD: Diagrama de Contexto



Fuente: Juan Antonio López Quesada,

DFD: Diagrama 0

0. Sistema de pedidos



Fuente: Juan Antonio López Quesada,

Diccionario de Datos

- Se utiliza notación similar a BNF

Símbolo	Significado
<code>:</code>	Compuesto de
<code>+</code>	Concatenación
<code>()</code>	Datos opcionales
<code>{n}</code>	N iteraciones
<code>[...]</code>	Selección de una alternativa
<code>* ... *</code>	Delimita comentarios

- Ejemplo:

- Teléfono:= [local | móvil | nacional | extranjero]
- local := {dígito}8
- nacional:= '01'+{dígito}10
- móvil := '044'+{dígito}10
- extranjero := '001'+{dígito}10
- dígito := ['0'|'1'|'2'|'3'|'4'|'5'|'6'|'7'|'8'|'9']

Fuente: Juan de Lara, EPS, UAM

81

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM



Descripción de procesos (lenguaje estructurado)

SI la factura excede de \$3,000

SI la cuenta del cliente tiene alguna factura sin pagar más de 60 días

Dejar la confirmación pendiente de este pago.

SI NO (la cuenta está en buen estado)

Hacer confirmación y factura

FIN-SI

SI NO (la factura es de \$3,000 o menos)

SI la cuenta del cliente tiene alguna factura sin pagar más de 60 días
Hacer la confirmación, la factura y escribir un mensaje sobre informe de crédito

SI NO (la cuenta está en buen estado)

Hacer confirmación y factura

FIN-SI

FIN-SI

Fuente: Juan Antonio López Quesada,

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM



82

Casos de Uso

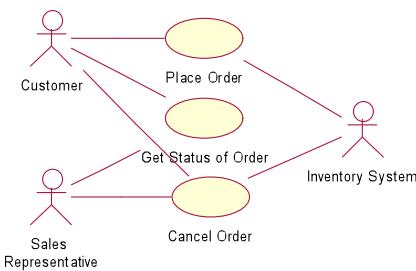
- Nombre del Caso de Uso: Colocación de orden
- Descripción corta:
 - El cliente proporciona el domicilio y una lista de los números de artículo
 - El sistema confirma la orden
- Flujo básico de eventos
 - Cliente digita su nombre y domicilio
 - Cliente digita los # de artículo que desea ordenar
 - El sistema le provee la descripción y precio de cada artículo
 - El sistema continuará generando la información para cada artículo digitado
 - El cliente captura la información de la tarjeta de crédito
 - El sistema valida la información de la tarjeta de crédito
 - El sistema genera un recibo al cliente



83

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Casos de Uso



84

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Descripción de procesos (tabla de decisión)

ESTADO DE LA CUENTA	CORRECTO	IMPAGADO	CORRECTO	IMPAGADO
NETO-FACTURA	>3000	>3000	<=3000	<=3000
CONFIRMACIÓN PENDIENTE		X		
HACER CONFIRMACIÓN	X		X	X
HACER FACTURA	X		X	X
ESCRIBIR MENSAJE				X

Fuente: Juan Antonio López Quesada,

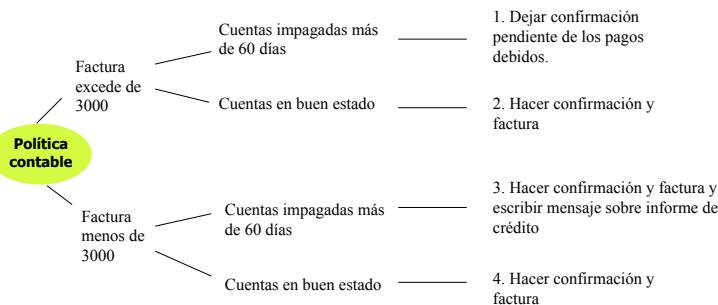
85

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Mantenimiento de sistemas

- Definición: **Modificación de un producto de software, después de su entrega, para corregir errores, mejorar el rendimiento u otros atributos, o adaptar el producto a cambios del entorno.** IEEE Std. 1219-1998
- La complejidad del mantenimiento de un sistema de software:**
 - Consumo muchos recursos, El mantenimiento consume más del 60% del coste de todo el ciclo de vida
 - El sistema ya está en uso. Las actividades de mantenimiento resultan muy visibles para el cliente.
 - Pueden afectar negativamente a la imagen de la organización.
 - Es una actividad crítica generalmente subvaluada.

Descripción de procesos (Árbol de Decisión)



Fuente: Juan Antonio López Quesada,

86

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Mantenimiento de sistemas

- El mantenimiento del software se clasifica generalmente en cuatro categorías, en función de cuál es la causa que motiva el cambio:**
 - Adaptativo
 - Correctivo
 - Perfectivo
 - Preventivo

Mantenimiento de sistemas

- **Correctivo:** Orientado a la corrección de defectos
- **Adaptación:** Modificación para migrar el software a otro ambiente operativo
- **Mejora:** Para ampliar los requerimientos funcionales originales o iniciales
- **Preventivo.** Ejecutar cambios en el software para facilitar su corrección, adaptación y mejora



89

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Mantenimiento de sistemas

- **Adaptativo:** Permite al software continuar en funcionamiento, adaptándose a cambios producidos en su entorno de operación.
- Los cambios típicos se suelen centrar en el hardware con el que interactúa, en el sistema operativo, o en formatos de datos que recibe o envía.
- **Correctivo:** Tiene como finalidad corregir fallos o problemas. Dentro del mantenimiento correctivo se suele diferenciar entre “de emergencia” o “de agenda”; en función de la urgencia con la que deba aplicarse la solución.



90

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Mantenimiento de sistemas

- **Perfectiva:** Se realiza para dar respuesta a nuevos requisitos del cliente, o para mejorar el rendimiento del sistema.
- **Preventivo:** En su versión de 1993, el estándar IEEE 1229 incluía también en su clasificación el mantenimiento preventivo como aquel que se realiza para evitar la aparición futura de errores, o para mejorar la integridad de producto en prevención de éstos. Algunos textos lo consideran como un 4º tipo de mantenimiento, y otros lo incluyen como mantenimiento correctivo.



91

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Mantenimiento de sistemas

Las 7 fases del mantenimiento:

1. **Identificación, clasificación y priorización del problema o de la modificación.** Cada petición de cambio debe identificarse, clasificarse y asignarle una prioridad, teniendo en cuenta qué tipo de mantenimiento implica (correctivo, adaptativo, perfectivo y si es de emergencia)
2. **Análisis.** La fase de análisis emplea la relación de peticiones de cambio registradas y validadas para analizar su viabilidad, alcance de las modificaciones y preparar un plan preliminar de diseño implementación y entrega
3. **Diseño.** En esta fase se emplea toda la documentación del sistema, del proyecto y la generada en la fase anterior (análisis) para diseñar la modificación del sistema.
4. **Implementación.** A partir del diseño realizado y verificado, el código y la documentación del sistema y del proyecto se lleva a cabo el trabajo de implementación.
5. **Pruebas de sistema y de regresión.** Tras la implementación deben realizarse las pruebas del sistema modificado. Las pruebas de regresión son parte de las pruebas del sistema que comprueban que el código modificado no ha introducido errores nuevos.
6. **Pruebas de aceptación.** Sobre el sistema completamente integrado, el cliente, los usuarios o un tercero nombrado por el cliente lleva a cabo las pruebas de aceptación
7. **Entrega.** Entrega del sistema modificado.



92

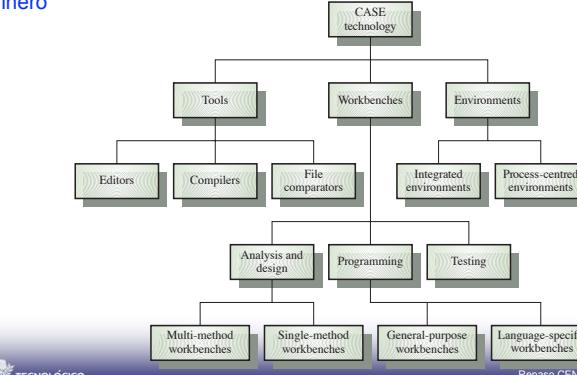
Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Computer Aided Software Engineering (CASE)

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Métodos y Herramientas

Diversas aplicaciones destinadas a **aumentar la productividad** en el desarrollo de software, **reduciendo el costo** de las mismas en términos de **tiempo y de dinero**



Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Herramientas CASE

- Database Management
- Analysis & Design
- Interface Design & Development
- Prototyping
- Programming
- Web Development

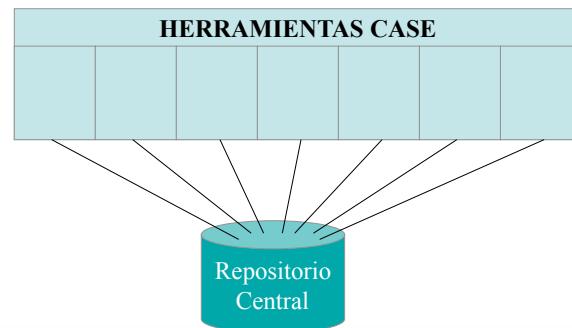
Herramientas CASE

- Business Process Engineering
- Process Modeling & Management
- Project Planning
- Project Management
- Risk Analysis
- Requirements Tracing
- Metrics Management
- Documentation Management

Herramientas CASE

- Quality Assurance
- Integration & Testing
- Test Management
- System Software
- Software Configuration Management
- Code Reengineering

Integrated CASE (i-CASE)



Técnicas de Estimación del Software

COCOMO

- COCOMO es el modelo de estimación más utilizado actualmente
- Fue desarrollado por Barry Boehm en 1981
- COCOMO predice el esfuerzo y el calendario para el desarrollo de un producto de software, basado en datos como el tamaño del software y un conjunto de habilitadores (drivers) que impactan en la productividad

COCOMO: Tres modelos

- COCOMO tiene tres representaciones o modelos clasificados por su complejidad:
 - El modelo Básico
 - El modelo Intermedio
 - Y el modelo Detallado

Las características del proyecto y el producto a desarrollar

- **Modo Orgánico**
 - Se desarrolla en un entorno conocido y estable
 - Similar a otros proyectos desarrollados previamente
 - Relativamente pequeño y no requiere innovaciones
- **Modelo Semi-desprendido**
 - Intermedio entre el modelo orgánico y el embebido
- **Modelo embebido**
 - Inflexible en sus restricciones y requerimientos
 - El producto requiere de mucha innovación

COCOMO: Supuestos

- El principal factor crítico es el # de Instrucciones fuente: DSI: Delivered Source Instructions
- Las estimaciones de COCOMO suponen que el proyecto será bien administrado
- Se asume que la especificación de requerimientos no se altera sustancialmente después de las etapas de planeación e ingeniería de requerimientos

Modelo Básico de COCOMO

- El modelo Básico de COCOMO estima el esfuerzo de desarrollo utilizando como única variable predictiva : DSI y los tres modelos de desarrollo

Ecuaciones del modelo COCOMO

Mode	Effort	Schedule
Organic	$E=2.4*(KDSI)^{1.05}$	$TDEV=2.5*(E)^{0.38}$
Semidetached	$E=3.0*(KDSI)^{1.12}$	$TDEV=2.5*(E)^{0.35}$
Embedded	$E=3.6*(KDSI)^{1.20}$	$TDEV=2.5*(E)^{0.32}$

Método de Puntos de Función

- Es una técnica estructurada que clasifica los componentes de un sistema
- Es un método para particionar un sistema en componentes más pequeños de modo que puedan ser entendidos y analizados
- Mide el software cuantificando la funcionalidad proveída por el usuario en la etapa de diseño lógico
- La funcionalidad lógica desde la perspectiva de un usuario “sofisticado”
- Es un método estándar para medir el desarrollo del software desde el punto de vista del usuario

Ejemplo del modelo Básico

- Asumamos que el DSI es de 25,000 LOC

- El modelo de producto es orgánico

$$E = 2.4 * KDSI^{1.05} \quad TDEV = 2.5 * E^{0.38}$$

$$E = 2.4 * 25^{1.05} \quad TDEV =$$

$$E =$$

Puntos de Función

- Basado en la combinación de las siguientes características de un programa
 - Entradas y Salidas Externas
 - Interacciones con el usuario: Interfases
 - Interfases externas
 - Archivos utilizados por el sistema
- Se define un peso o factor de ajustes para cada uno de los componentes

$$UFC = \text{Sumatoria}(\# \text{ de componentes de cada tipo}) * (\text{factor})$$

Puntos de Función

- La cantidad de puntos de función se modifica dependiendo de la complejidad del proyecto
- Los PF se pueden utilizar para estimar el LOC tomando en cuenta tablas estándar de promedio de LOC por PF para un lenguaje de programación específico
 - LOC = AVC * # de PF
 - AVC es un factor dependiente del lenguaje que va de 200 a 300 para lenguaje ensamblador hasta 2 a 40 para un 4GL



109

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Puntos de Función

- Sistemas de tiempo real: 40 a 160 LOC/ Programador-Mes
- Software de sistema: 150 a 400 LOC/ Programador - Mes
- Software de negocios: 200 a 900 LOC/ Programador-Mes



110

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM



Administración de Proyectos de Software

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

¿Qué es la Administración de Proyectos?

- La Administración de Proyectos es "la aplicación de los conocimientos, habilidades, herramientas y técnicas a las actividades del proyecto para satisfacer las necesidades de los proyectos." *



Ing. Jakeline Marcos Abed

• (PMBOK
Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM)

New Top Ten Factors for IT Project Success

Rank	Success Factor
1	Executive Support
2	User Involvement
3	Experienced Project Manager
4	Clear Business Objectives
5	Minimized Scope
6	Standard Software Infrastructure
7	Firm Basic Requirements
8	Formal Methodology
9	Reliable Estimates
10	Other

• Table 1.3 Source: *Extreme Chaos*. The Standish Group International, Inc. 2001.

• http://www.standishgroup.com/sample_research/index.php

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM



Ventajas de utilizar una Administración de Proyectos formal

- Mejor control de los recursos físicos, humanos y financieros
- Mejor relación con los clientes.
- Menor tiempo de desarrollo.
- Menores costos.
- Mayor calidad y confiabilidad.
- Mayores márgenes de utilidad.
- Mayor productividad.
- Mejor coordinación interna.
- Eleva la moral de los empleados (menos estrés).



Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Mayor probabilidad de éxito con:

- Enfoque socio-técnico
- Enfoque de Admon. De Proyectos:
 - Procesos e infraestructura (Metodología)
 - recursos
 - expectativas
 - competencia
 - Eficiencia y efectividad
- Enfoque de Admon. del Conocimiento:
 - Lecciones aprendidas, mejores prácticas y compartir el conocimiento.



Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

¿Qué es un Proyecto?

- Un **proyecto** es “un **esfuerzo temporal** realizado para crear un producto único o un servicio.”
- Un proyecto termina cuando sus **objetivos** se han **alcanzado**, o el proyecto se haya terminado.
- Los proyectos pueden ser grandes o pequeños y tener un corto o largo **tiempo** para completarse.

•(PMBOK® Guide)



Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Atributos de un proyecto

- Un proyecto:
 - Tiene un **propósito único que provee valor**.
 - Es **temporal**.
 - Sentido de pertenencia.
 - Requiere **recursos**.
 - Se desarrolla de manera **progresiva**.
 - Tiene actividades y tareas **interdependientes**.
 - Requiere recursos de **varias áreas**.
 - Debe tener un **cliente principal o patrocinador**.
 - Involucra **incertidumbre y riesgos**

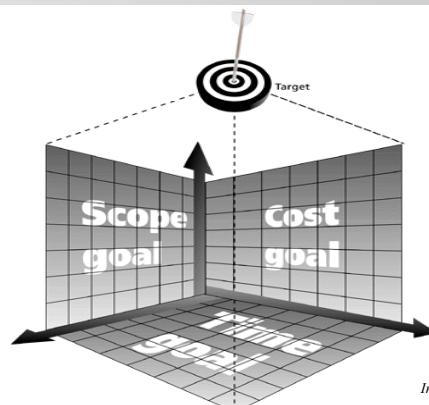


Triple limitación

- Cada proyecto está limitado por:
 - **Alcance**: ¿Qué trabajo se va a hacer?
 - **Tiempo**: ¿Cuánto se debe de tardar?
 - **Costo**: ¿Cuánto debe de costar?
- El administrador de proyectos debe balancear estas tres variables.



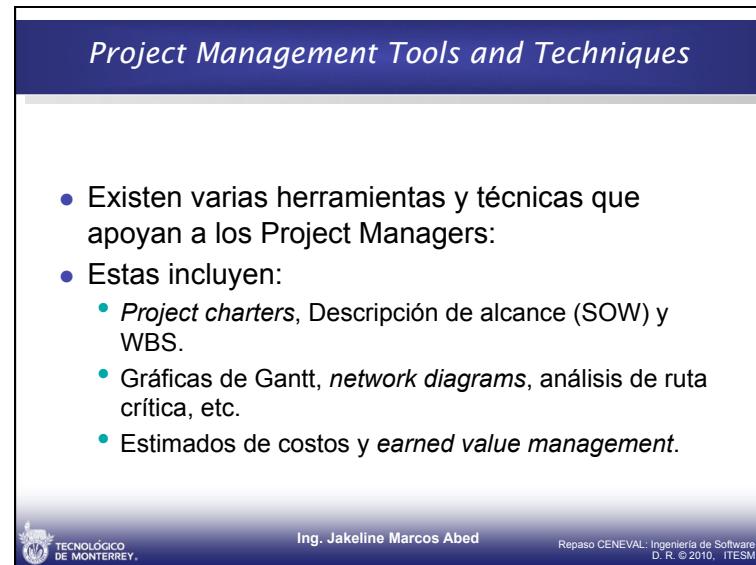
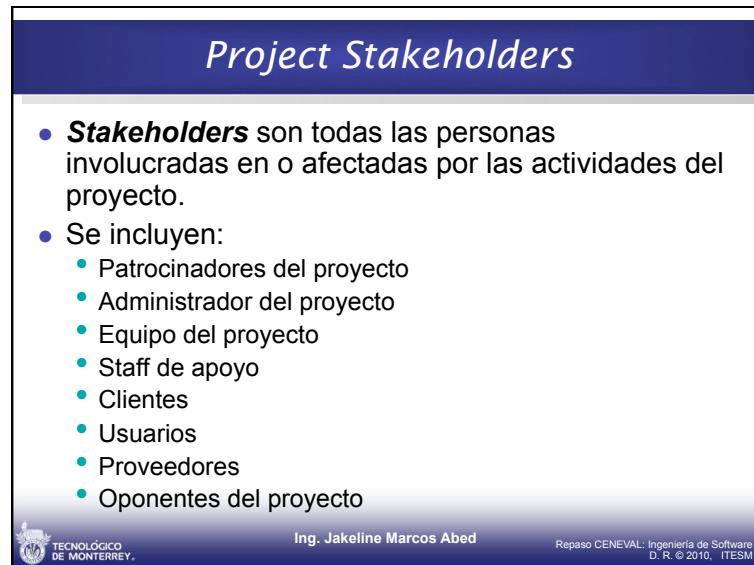
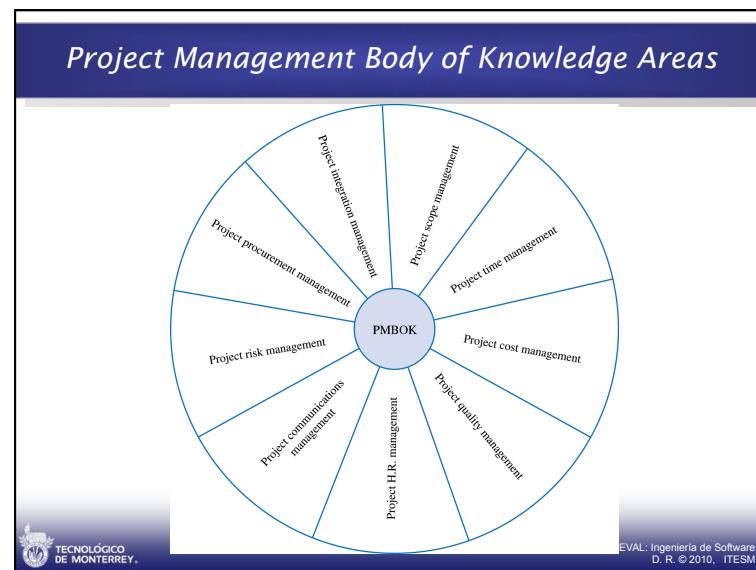
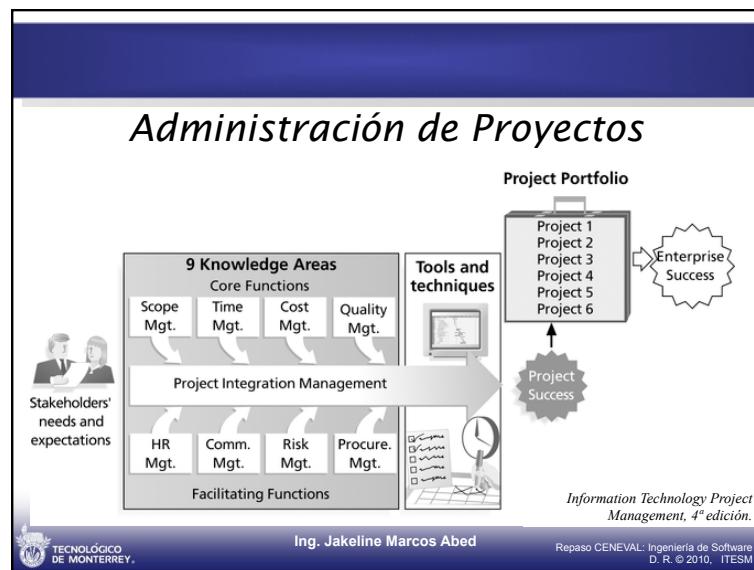
Triple limitación



9 Project Management Knowledge Areas:

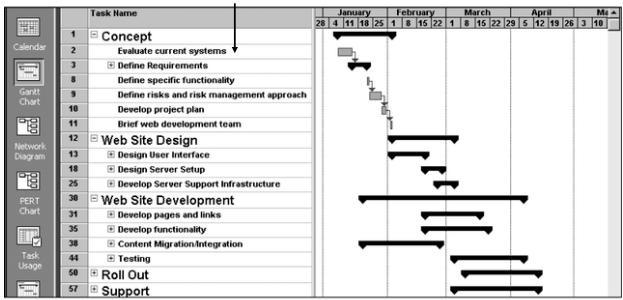
- **Knowledge areas** describen las competencias claves que los administradores de proyectos deben desarrollar.
 - 4 referentes a **objetivos específicos del proyecto** (alcance, tiempo, costo, y calidad).
 - 4 referentes a los **medios** por medio de los cuales se alcanzan los objetivos del proyecto (recursos humanos, comunicación, riesgos, y gestión de las adquisiciones o *aprovisionamiento*).
 - **Gestión de integración**, afecta y es afectada por todas las demás áreas.





Ejemplo de Gráfica de Gantt

- Work Breakdown Structure (WBS)

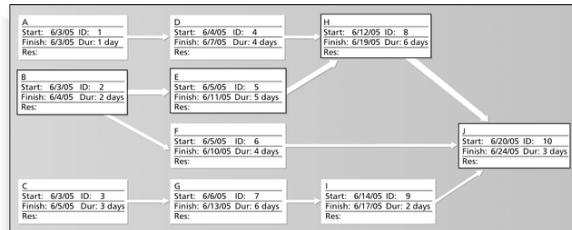


El WBS se muestra a la izq., y cada fecha de inicio y terminación se muestra en la derecha. La primera vez que se usó fue en 1917, y obviamente las primeras gráficas fueron dibujadas a mano.



Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Ejemplo de Network Diagram



Cada cajita es una tarea del WBS. Las flechas muestran las dependencias entre las tareas. Las que tienen recuadro muestran la **ruta crítica**. Secuencia más larga de actividades en el menor tiempo posible para completar el proyecto. Si una tarea de la ruta crítica se tarda más de lo planeado, todo el proyecto se retrasa, a menos que se haga algo !. Inicialmente fueron usados en 1958 en el proyecto Navy Polaris antes de que existieran los conceptos de PM.



Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Dimensiones de la gestión de proyectos

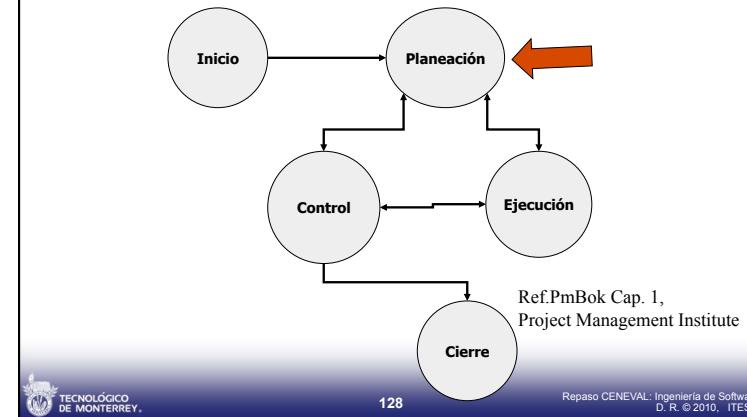
- Personas: Roles y competencias
- Producto: objetivos y alcance
- Procesos: Metodologías y herramientas
- Proyecto: Metodologías y herramientas



127

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Ciclo de Vida de Proyectos



128

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Gerencia de Proyectos (áreas de conocimiento y objetivos)			
5. Administración del Alcance del Proyecto	6. Administración del Programa del Proyecto	7. Administración del Presupuesto	8. Administración de la Calidad del Proyecto
Incluye los procesos requeridos para asegurar que el proyecto incluya todo el trabajo requerido, y solo el trabajo requerido, para terminar el proyecto exitosamente.	Incluye los procesos requeridos para asegurar la terminación a tiempo del proyecto.	Incluye los procesos requeridos para asegurar que el proyecto termine dentro del presupuesto aprobado.	Incluye los procesos requeridos para asegurar que el proyecto satisfaga las necesidades para el cual fué emprendido.
9. Administración de los Recursos Humanos	10. Administración de las Comunicaciones	11. Administración del Riesgo del Proyecto	12. Administración de los Abastecimientos
Incluye los procesos requeridos para lograr el uso más efectivo de las personas involucradas en el proyecto.	Incluye los procesos requeridos para asegurar la puntual y apropiada generación, recolección, disseminación, archivo, y última disposición de la información del proyecto.	Incluye los procesos concernientes con la identificación, análisis, y respuesta al riesgo del proyecto. Incluye el maximizar los resultados de eventos positivos y minimizar las consecuencias de eventos adversos.	Incluye los procesos requeridos para adquirir bienes y servicios externos a la organización a cargo del proyecto.

Administración de Proyectos de SW

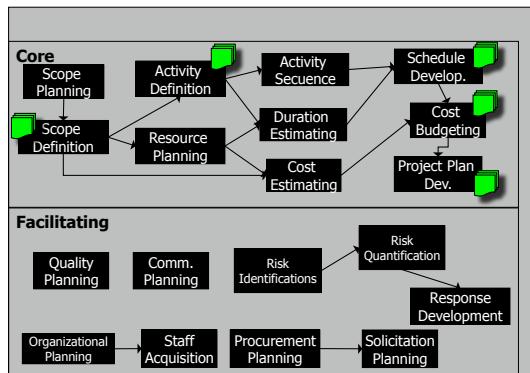
- Relativo a las actividades que aseguren que el SW es entregado a tiempo, dentro del presupuesto y con los requerimientos definidos por la organización que lo solicita
- Se requiere aplicar la administración de proyectos debido a un entorno restringido en presupuesto y calendario para el desarrollo del software

Características de la administración del software	
<ul style="list-style-type: none"> • El producto es intangible • El producto es dinámico y único • La ingeniería de software no se reconoce como una disciplina como la ingeniería química, mecánica o industrial • El proceso de desarrollo de software no es aplicado de manera estándar en las organizaciones 	

Actividades de administración

- Desarrollo de la propuesta
- Planeación y calendarización del proyecto
- Costeo y presupuestación del proyecto
- Monitoreo y control del avance
- Selección y evaluación del personal
- Preparar los reportes de avance

Fase de Planeación: PmBok



Estructura de un plan de proyectos

- Introducción: Objetivo y Alcance
- Organización del equipo de trabajo
- Metodología de trabajo: Ciclo de Vida
- Análisis de riesgos
- Requerimientos de HW y SW
- WBS
- Calendario: Gráfica de Gannt, Pert, CPM, etc
- Mecanismos de control y reporteo del avance

El Plan del Proyecto

- El plan del proyecto genera:

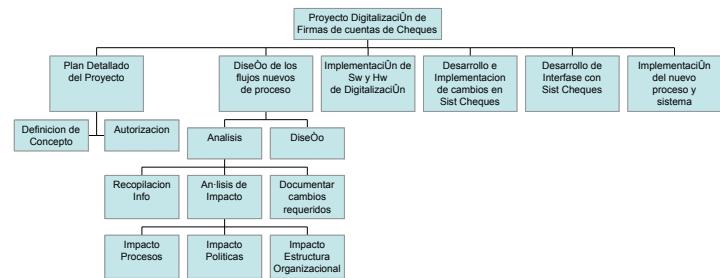
- La lista de recursos requeridos (humanos, tecnológicos, económicos, instalaciones, equipamiento, etc.): SDP
- La descripción detallada de las tareas : WBS
- La estimación de esfuerzo: horas-persona
- El calendario del proyecto

Calendarización del Proyecto: Scheduling

- Descomponer el proyecto en tareas y estimar el tiempo y los recursos requeridos para ejecutar dicha tarea
- Organizar las tareas en forma concurrente para optimizar el tiempo del equipo de trabajo
- Minimizar la dependencia entre las tareas para evitar retrasos o cuellos de botella
- Depende de la intuición y experiencia del AP

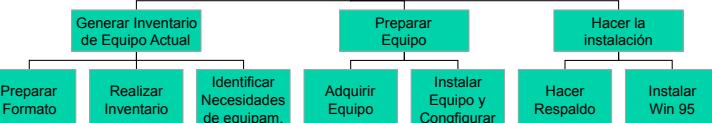
WBS Proyecto Digitalización de firmas

Proyecto Imagen

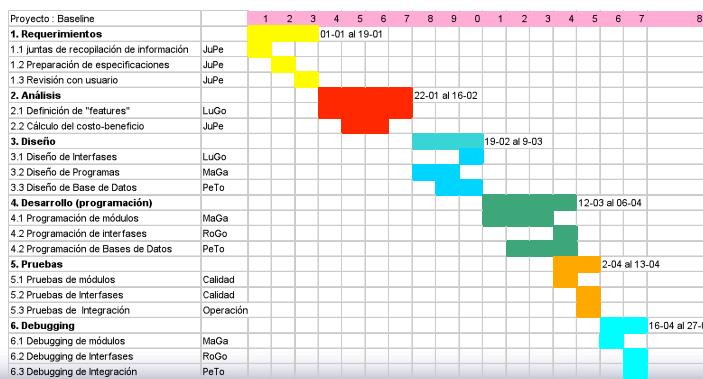


WBS: Migración a Win 95

Proyecto Migración win95

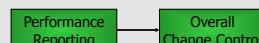


Plan Maestro de Trabajo



PmBok:Fase de Control

Core Processes



Facilitating



El enfoque en la Administración de Proyectos

- Control: es el comparar los avances/ resultados con respecto a los planes, de manera que se tomen acciones correctivas, cuando haya desviaciones
- Control = f (planes, resultados, información, motivación)
- No es un enfoque orientado al ejercicio del poder

Los niveles del control

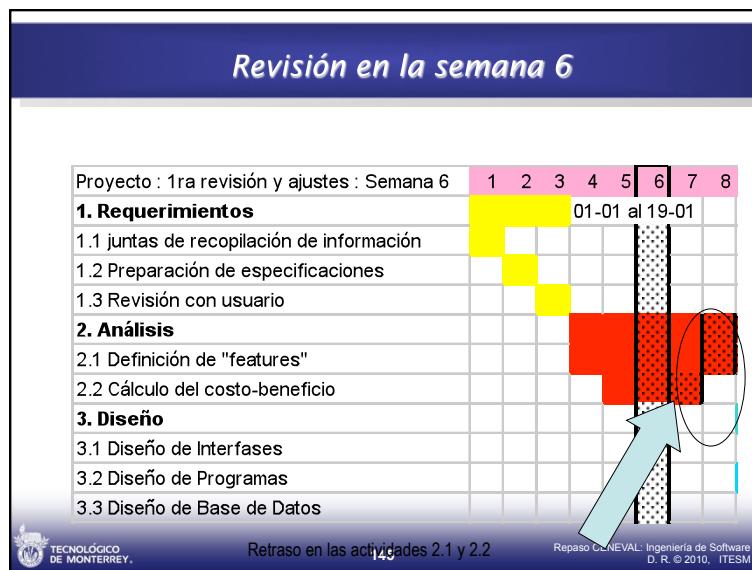
- Mega-control: Sobre un programa o grupo de proyectos
- Macro-Control: el que debe ejercer el líder del proyecto, concentrándose en los primeros niveles del WBS (programa, proyecto, tarea)
- Micro-control: el que debe ejercer cada persona del equipo del trabajo, a partir del plan estratégico
- **Cuidado: que el líder no caiga en la micro-administración...**

Proyecto Tipo: Calendario Inicial

Proyecto : Baseline	1	2	3	4	5	6	7	8	9	0
1. Requerimientos				01-01 al 19-01						
1.1 Juntas de recopilación de información										
1.2 Preparación de especificaciones										
1.3 Revisión con usuario										
2. Análisis									22-01 al 16-	
2.1 Definición de "features"										
2.2 Cálculo del costo-beneficio										
3. Diseño										
3.1 Diseño de Interfases										
3.2 Diseño de Programas										
3.3 Diseño de Base de Datos										

Reporte de Avance de Proyectos: Inicio

Es un proyecto con duración de 17 Semanas y un esfuerzo total de 18 semanas (Programación y Pruebas son en Paralelo)								
PM RESPONSABLE	GRUPO			STATUS	Iniciando el Proyecto			
APLICACION	Versatility			ESTIMADO	REAL			
ETAPA	SEM. COMPROM ISO	SEM. ACUMULADAS	SEM. REALES	SEM. ACUMULADAS	Fecha Inicio	Fecha Fin	Fecha Inicio	Fecha Fin
Requerimientos	3	3			01-Ene	19-Ene		
Análisis	4	7			22-Ene	16-Feb		
Diseño	3	10			19-Feb	09-Mar		
Programación/Pruebas	5	15			12-Mar	13-Apr		
Debugging	2	17			16-Apr	27-Apr		



1er. Reporte de Avance: semana 6

APLICACION	GRUPO			STATUS	1ra revisión de avance			
	Versatilidad				ESTIMADO	REAL		
ETAPA	SEM. COMPROMISO	SEM. ACUMULADAS	SEM. REALES Nuevas	SEM. REALES ACUMULADAS	Fecha Inicio	Fecha Fin	Fecha Inicio	Fecha Fin
Requerimientos	3	3	3	3	01-Ene	19-Ene	01-Ene	19-Ene
Análisis	4	7	5	8	22-Ene	16-Feb	22-Ene	23-Feb
Diseño	3	10	3	11	19-Feb	09-Mar	26-Feb	16-Mar
Programación/Pruebas	5	15	5	16	12-Mar	13-Abr	19-Mar	20-Apr
Debugging	2	17	2	18	16-Abr	27-Abr	23-Abr	04-May

Actividad 2.1 : estatus : No Terminada, Tiempo Requerido para concluir: 2 semanas (1+ de la comprometida)

Actividad 2.1 Avance Comprometido (Sem 6) = Semanas aplicadas / semanas totales = 3/4 = 75% avance comprometido

Avance Real (Sem 6) = Semanas aplicadas / semanas reales nuevas = 3 / 5 = 60% avance real actividad:

Retraso actividad : en % = 75% - 60% = 15%, en semanas = 5 semanas - 4 semanas = 1 semana

Retraso en % = Retraso total (semanas) / semanas totales comprometidas = 1 semana/17 Semanas = 5,9%

146

Rapso CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

TECNOLOGICO DE MONTERREY.

Reporte Directivo

Retraso del Proyecto: Semanas reales nuevas Totales - Semanas Comprometidas Totales = 18 semanas - 17 semanas = 1 semana

Retraso en % = Retraso total (semanas) / semanas totales comprometidas = 1 semana/17 Semanas = 5,9%

Reporte al Directivo
Proyecto : X, Líder del Proyecto : Juan X
Status a la semana: 6
Actividad en curso: Análisis, Responsable de la etapa: Jorge
Retraso en % de proyecto: 5.9%
Retraso en tiempo del proyecto: 1 semana
Fecha estimada Comprometida de finiquito: 27-Abril
Fecha estimada de finiquito nueva: 4 de Mayo
Observaciones: en el proceso de análisis se requiere una semana adicional para definir la funcionalidad del proyecto

147

Rapso CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

TECNOLOGICO DE MONTERREY.

- ## Gestión de Riesgos
- Gestión de riesgos se refiere a la identificación de los riesgos y la definición de acciones para mitigar su efecto en un proyecto
 - Un riesgo es la probabilidad de que una situación adversa pueda ocurrir
 - Un riesgo en **proyectos** puede afectar el **calendario** y/o los recursos
 - Un riesgo de **producto** puede afectar la **calidad** o desempeño del software que se está desarrollando
 - Un riesgo del **negocio** puede afectar al equipo de trabajo que está desarrollando o facilitando el software
- 148
- Rapso CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM
- TECNOLOGICO DE MONTERREY.

Algunos riesgos en proyectos de desarrollo

Risk	Affects	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organisational management with different priorities.
Hardware unavailability	Project	Hardware that is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule.
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool under-performance	Product	CASE tools which support the project do not perform as anticipated.
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

El proceso de gestión de riesgos

• Identificación del riesgo

- Identificar los riesgos del proyecto, producto y negocio

• Análisis de riesgos

- Evaluar la probabilidad y consecuencias de estos riesgos

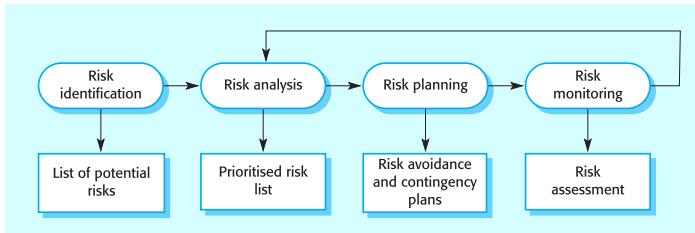
• Planeación de riesgos

- Generar las acciones para evitar o minimizar el efecto de los riesgos

• Monitoreo de los riesgos

- Dar seguimiento a los riesgos durante la vida del proyecto

El proceso de gestión de riesgos



Análisis de Riesgos

• Evaluar la probabilidad y seriedad de cada riesgo

• La probabilidad puede ser muy baja, baja, moderada, alta o muy alta

• Los efectos de los riesgos pueden ser: catastrófico, serio, tolerable o insignificante

Estimación de la magnitud de la pérdida

- Se maneja bajo una tabla donde se relaciona la magnitud (tiempo) y la probabilidad de ocurrencia.

Riesgo	Probabilidad	Magnitud	Exposición	Pri
Plan. Optimista	50%	5	2.5	2
Retraso del Personal externo	25%	4	1.0	3
Cambios en los requerimientos	40%	8	3.2	1



153

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Monitoreo del Riesgo

- Evalúe de manera continua los riesgos identificados y determine si hay cambios en la probabilidad de ocurrencia y el impacto del mismo
- No olvide integrar a la agenda de juntas de avance, el análisis de riesgos
- Considere que durante el desarrollo del proyecto pueden aparecer otros riesgos potenciales



155

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Planeación de riesgos

- Considera a cada riesgo y desarrolle la estrategia para administrar cada uno
- Estrategias de eliminación:
 - La probabilidad de ocurrencia puede ser reducida
- Estrategias de minimización
 - El impacto del riesgo puede ser minimizado
- Planes de contingencia
 - Si aparece el riesgo, los planes de contingencia son acciones para eliminar o reducir el impacto



154

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Calidad en el Software

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Calidad en el Software

- ¿Cómo medimos la calidad en el software entregado?

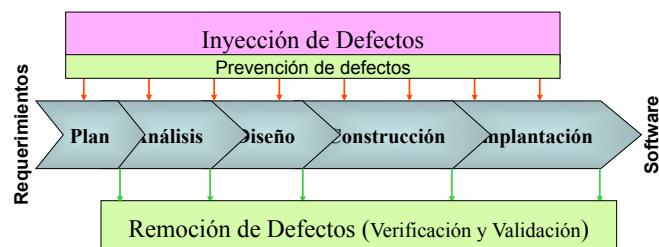
- Existen varios “factores de la calidad”



Calidad en el Software

- Calidad SW = confiabilidad
 - = densidad de defectos entregados (defectos / KLOC)
- Defecto = inconsistencia con requerimientos
 - Defecto ≠ Error

Necesitamos: eliminar los defectos



La calidad del producto final depende de la calidad imprimida en las diferentes tareas, actividades y procesos.

Calidad en el Software

- Actividades para
 1. Remover los defectos inyectados
 2. Prevenir la inyección de defectos
- Mientras más temprano mejor

Costos por arreglar defectos vs Fase

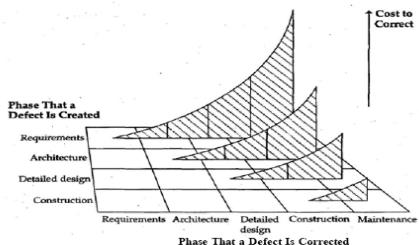


FIGURE 3.5 Increase in defect cost as time between defect creation and defect correction increases. Effective projects practice "phase containment"—the detection and correction of defects in the same phase in which they are created.



TECNOLÓGICO
DE MONTERREY.

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Calidad en el Software: Verificación y Validación vs SQA

Aseguramiento de la calidad

La función del Aseguramiento de la Calidad es garantizar que la organización realiza el trabajo conforme a los procedimientos y métodos establecidos para el proyecto.

IEEE Std. 730-1998

Relación de SQA con Verificación y validación.

El Aseguramiento de la calidad (SQA) es una metodología interna cuya principal finalidad es garantizar que el flujo del trabajo cumple con las normas que tiene impuestas el desarrollador (por su normativa interna, por imposición del cliente, etc.).

SQA no evalúa el producto producido en esa fase o en ese proyecto, sino el proceso que lo ha producido. No evalúa el producto, evalúa el proceso.

Validación y Verificación enfocan su análisis en atributos del producto generado.



TECNOLÓGICO
DE MONTERREY.

163

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Calidad en el Software: Verificación y Validación

Es la disciplina de gestión y actividad técnica para garantizar que el software operará según lo especificado en los **requisitos** y las **necesidades del usuario**, que se lleva a cabo a través de procesos de:

Verificación:

Determinación con medios objetivos y repetibles de que un elemento satisface los requisitos.

- ¿Se está construyendo adecuadamente el producto?
- La verificación se realiza “contra” el entorno de desarrollo o del proyecto.

Validación:

Determinación con medios objetivos y repetibles de que un elemento puede emplearse para el fin que tiene asignado.

- ¿Se está construyendo el producto adecuado?
- La validación se realiza “contra” el entorno cliente, donde el producto debe cumplir su finalidad.



TECNOLÓGICO
DE MONTERREY.

162

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Calidad en el Software: Verificación y Validación

Definiendo los objetivos

Las consideraciones que deben contemplarse para evaluar la planificación de las actividades de Validación y Verificación son:

- **Nivel de integridad del proyecto.**
Mide la “criticidad” del software. El estándar IEEE 1012 establece que el “plan de validación y verificación del software” (SVVP) debe especificar un método para clasificar el nivel de integridad del software de cada subsistema de software del proyecto.
Análisis de criticidad: Análisis de daños (*Hazard analysis*) Análisis de riesgos (*Risk Analysis*)
- **Mínimo de tareas recomendadas para el nivel de integridad del proyecto.**
La regulación interna de la organización desarrolladora puede determinar qué tareas de V&V deben realizarse para cada nivel de integridad.
El estándar IEEE 1012 define 4 niveles de integridad e incorpora una tabla en la que se estipulan las actividades mínimas de V&V en función del nivel.
- **Intensidad y rigor necesarios** en las tareas de Validación y verificación.
El nivel de integridad no sólo determina qué tareas deben realizarse, sino también su intensidad y rigor. Por ejemplo, si lo realiza el propio personal de desarrollo, otro equipo de desarrollo diferente, o incluso una organización externa (auditora).
- Los **criterios** que se emplearán en las tareas de V&V para establecer los **parámetros** mínimos de corrección, consistencia, precisión



TECNOLÓGICO
DE MONTERREY.

Fuente: Juan Palacio

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

164

Calidad en el Software: Verificación y Validación

•Análisis de criticidad: análisis de daños.

La definición formal del análisis de daños (a nivel de sistema) es:

"Análisis de fuentes potenciales de daños o de situaciones que pueden generar daños en términos de daños a personas, daños a la salud, o al entorno, o una combinación de ellos".

IEC 60300-3-9, 1995

No obstante el estándar para validación y verificación IEEE 1012-1998 da una definición más amplia que incluye también pérdidas económicas, fallo en la misión del sistema, o impacto social adverso. Para nosotros "daño" en el marco de validación y verificación de software incluye por tanto:

- Daños a las personas.
- Daños al medio ambiente.
- Pérdidas económicas.
- Fallo en la finalidad del sistema.
- Impacto social adverso.

Para realizar el análisis de daños deben **identificarse las consecuencias que pueden ocasionar los fallos en el software**. Es posible que no generen daños físicos, pero si en términos de pérdidas económicas (para el desarrollador, para el cliente, para los usuarios), o de impacto social adverso (desprestigio del cliente, del desarrollador, de los usuarios, de terceros).

Riesgo: probabilidad de que se produzca un daño identificado



TECNOLÓGICO
DE MONTERREY.

165

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Calidad en el Software: Verificación y Validación

Análisis de criticidad: análisis de riesgos. Riesgo: probabilidad de que se produzca un daño identificado.

•NATURALEZA DEL RIESGO	•CAUSAS TÍPICAS
•Propios del sistema	•Los identificados en el análisis de daños
•Propios del desarrollo de software	<ul style="list-style-type: none">▪ Complejidad innecesaria▪ Complejidad intrínseca del diseño mayor de la necesaria▪ Baja calidad▪ Incumplimiento de estándares necesarios▪ Inestabilidad de los requisitos▪ Problemas con herramientas y métodos▪ Inestabilidad, bugs en compiladores, etc.▪ Comportamiento imprevisto de los interfaces▪ Interfaces con hardw. y softw. externo en la implementac.▪ Inestabilidad y cambio rápido de las plataformas tecnológicas
•Propios de los desarrollos por proyecto ^[1]	<ul style="list-style-type: none">▪ Presión en costes y agendas.▪ Lagunas en planificación y gestión.▪ Retrasos en subcontrataciones.

•Fuente: Juan Palacio



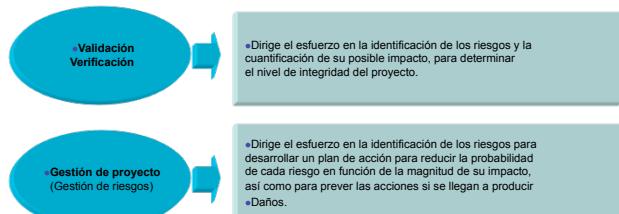
[1] En los proyectos de software se suelen dar con mayor intensidad los riesgos típicos indicados.

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Calidad en el Software: Verificación y Validación

•Análisis de criticidad: análisis de riesgos.

Validación y Verificación no gestionan las causas de los riesgos. Esa gestión pertenece a la "gestión de riesgos", dentro de la "gestión del proyecto".



•Fuente: Juan Palacio



TECNOLÓGICO
DE MONTERREY.

167

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Calidad en el Software: Verificación y Validación

•Niveles de integridad: Una vez realizado el análisis de criticidad a través de los análisis de daños y de riesgos, resulta posible establecer el nivel de integridad del proyecto y adecuar a él las tareas de validación y verificación. El estándar IEEE 1012-1998 define 4 niveles de integridad:

•Nivel	•Dimensión del daño por fallo del Software	•Mitigación aplicable
•4	<ul style="list-style-type: none">▪ Pérdida de vida▪ Pérdida del sistema▪ Graves pérdidas económicas o sociales	•No es posible mitigar los daños producidos
•3	<ul style="list-style-type: none">▪ El sistema no realiza el fin previsto ni en todo ni en parte.▪ Graves pérdidas económicas o sociales	•Es posible una mitigación parcial de los daños producidos
•2	<ul style="list-style-type: none">▪ No se pueden realizar funcionalidades parciales del sistema.▪ Pérdidas económicas o sociales importantes	•Se pueden mitigar los daños producidos
•1	<ul style="list-style-type: none">▪ Una determinada funcionalidad del sistema no se realiza.▪ Consecuencias mínimas	•No es necesario mitigar los daños

El modelo del estándar resulta válido, pero cualquier modelo, adecuado a las circunstancias del sistema y el entorno de desarrollo, puede resultar igualmente válido.

•Fuente: Juan Palacio



Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

168

Calidad en el Software.

Verificación y Validación: en el ciclo de vida del SW

REQUERIMIENTOS:

En esta fase, la verificación y validación comprueba el principal producto generado en esta fase: la especificación de requisitos del software.

Se analizan las propiedades de calidad del Documento:

- Completo, correcto, consistente, modificable, trazable
- Posibles, necesarios, priorizados, correctos, verificables.

DISEÑO

Comprobación de que el diseño realizado comprende todos los requisitos sin omisiones y sin complejidad innecesaria. Los aspectos generales que se analizan son:

- Trazabilidad entre requisitos y diseño.
- No hay omisiones ni añadidos.
- El diseño es apropiado para los objetivos deseados del sistema.
- El diseño es conforme con los estándares, prácticas y convenciones acordadas para el proyecto
- El diseño es comprensible por el equipo de desarrollo y el posterior de mantenimiento.
- Contiene información suficiente para realizar las pruebas de unidad y de integración.
- La documentación está completa, incluyendo gráficas o especificaciones necesarias.

Calidad en el Software.

Verificación y Validación: en el ciclo de vida del SW

OPERACIÓN

Una vez instalado y puesto en servicio el sistema de software, la verificación y validación valora el impacto que los cambios pueden suponer en el nivel de integridad del sistema, o los riesgos o daños que pueden introducir.

Incluye la monitorización y evaluación del entorno de operación.

MANTENIMIENTO

Una vez puesto en servicio el sistema de software, las modificaciones del entorno, o la presencia de errores, o la necesidad de ampliar su funcionalidad requerirán emprender tareas de mantenimiento, que en esencia, y a menor escala son pequeñas acciones de desarrollo que pueden introducir modificaciones en el nivel de integridad, y requerir revisiones en los análisis de criticidad, daños y riesgos, así como requerir posteriores acciones de verificación y validación sobre las modificaciones de requisitos, diseño, desarrollo y pruebas.

Calidad en el Software.

Verificación y Validación: en el ciclo de vida del SW

IMPLEMENTACIÓN

La implementación transforma la descripción del diseño en componentes de que juntos integran el producto final de software. La labor de verificación y validación comprueba:

- Conformidad del código
 - Con las especificaciones del diseño
 - Con los estándares aplicables
 - La idoneidad del código para obtener el producto con el nivel de integridad deseado.

PRUEBAS

La verificación y validación de las pruebas garantiza que se han cumplido los requisitos del sistema y del software, alcanzando los niveles de integridad requeridos.

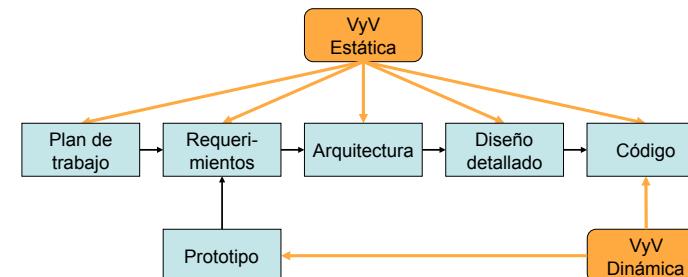
En sistemas con niveles de integridad 3 y 4 es necesario que el equipo de verificación y validación realice los planes y procesos de prueba, así como su ejecución.

Para niveles 1 y 2 es suficiente con verificar las pruebas realizadas por el equipo de desarrollo.

INSTALACIÓN

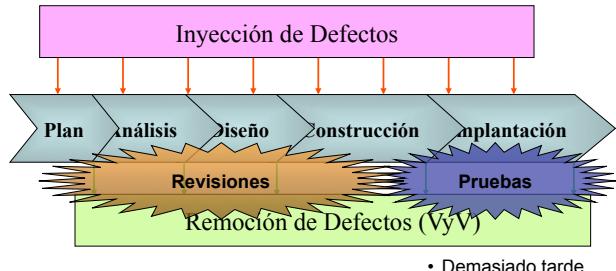
Se comprueba el rendimiento del sistema de software al ejecutarse en el entorno del cliente, así como que los procedimientos de instalación son correctos.

Tipos de Verificación y Validación



■ = Producto de Trabajo

Proceso de VyV



Revisiones de SW



Métricas: orientadas a tamaño

- Errores por KLOC
- Defectos por KLOC: Densidad de defectos
- Defectos corregidos/defectos encontrados
- Errores por persona-mes
- LOC por persona-mes
- PF por persona-mes
- \$ por KLOC

Métricas orientadas a funciones

Parámetro	#	Sencillo	Promedio	Complejo	Total
Inputs de usuario	3	4	6		
Outputs de usuario	4	5	7		
Consultas de usuario	3	4	6		
Archivos	7	10	15		
Interfases externas	5	7	10		

Tabla de LOC/PF

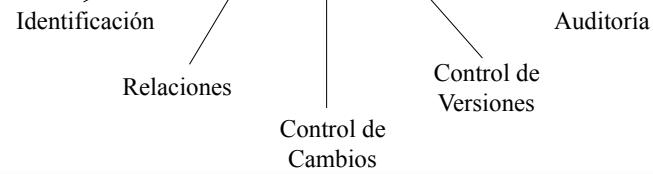
- | | |
|----------------|-----|
| • Ensamblador | 320 |
| • C | 128 |
| • C++ | 64 |
| • Visual Basic | 32 |
| • PowerBuilder | 16 |

Administración de la Configuración del Software

El software debe configurarse en forma correcta

Configuración del SW (Componentes)

- Programas
- Documentos
- Datos



Problemas:

- Actualización simultánea
 - Cuando dos o más programadores trabajan de forma separada en el mismo programa, el último que haga cambios puede fácilmente destruir el trabajo de otros.
- Código compartido
 - Cuando un error es corregido en código compartido por varios programadores, algunos de ellos no son notificados.
- Versiones
 - Muchos programas son desarrollados con "lanzamientos" evolutivos. Con una versión en uso por el cliente, otro en pruebas, y un tercero en desarrollo, la reparación de errores debe ser propagada entre ellos.

Propósito de la Admon. De Configuración

- El propósito de la Admon. De Configuración (CM) es **establecer y mantener la integridad de los productos de trabajo** utilizando la *identificación de la configuración, control de la configuración , contabilidad de los status de configuración, y auditorías de configuración.*



TECNOLÓGICO
DE MONTERREY.

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

El área de proceso de la Admon. De Configuración involucra lo siguiente:

- **Identificar la configuración** de productos de trabajo seleccionados que compongan las líneas de referencia en determinados puntos en el tiempo
- **Controlar cambios** a los *ítems* de configuración
- Proveer o **crear especificaciones** para construir productos de trabajo del sistema de administración de la configuración
- **Mantener la integridad** de las líneas de referencia
- Proveer **status y datos actuales** de la configuración a desarrolladores, usuarios finales, y clientes



Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

*Línea de referencia **Baselines***

- **La versión establecida, oficial y actual de un producto de configuración**
- **Sirve de punto de partida o referencia** para desarrollo subsecuente o cambios. (el primer doc. de req. aprobado, el primer lanzamiento de código)



TECNOLÓGICO
DE MONTERREY.

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Criterios para seleccionar un ítem de configuración

Ejemplos de criterios para seleccionar ítems de configuración al nivel apropiado de producto de trabajo incluyen las siguientes:

- Productos de trabajo que puedan ser **usados por dos o más grupos**
- Productos de trabajo que se espera que **cambien en el tiempo** ya sea por errores o por cambios en los requerimientos
- Productos de trabajo que sean **interdependientes** de manera que un cambio en uno demanda un cambio en los otros
- Productos de trabajo **críticos** para el proyecto



Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Funciones de la CM:

- 1) Identificación de la Configuración
- 2) Control de la Configuración
- 3) Administración de Cambios
- 4) Contabilidad de Status
- 5) Autentificación de la Configuración



TECNOLÓGICO
DE MONTERREY.

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Función 1: Identificación de la Configuración

- Este es el proceso de **identificar componentes** y la manera en que se combinan para hacer el producto (configuración)
- Es requerido en todo nivel de descomposición que puede ser usado, vendido, remplazado o probado independientemente
- **Mucha** identificación puede ser difícil de mantener mientras que **muy poca** causa problemas de productividad



Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

La Identificación de la Configuración incluye ...

- **Convenciones de nomenclatura** para cada versión de software y para cada versión de cada elemento de configuración
 - Nombres únicos para propósitos de identificación
 - Nivel de versión o cambio para distinguir diferentes versiones.



TECNOLÓGICO
DE MONTERREY.

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Función 2: Control de la Configuración

- Esta tarea gira en torno a tener una sola copia oficial del código (**Baseline**) que sirva de punto de referencia
- Sub-tareas incluyen:
 - **Administración de la promoción.** Se debe dar seguimiento de cuando el software está listo para “promover” a un nuevo nivel, qué procedimientos se siguen a medida que pasa por niveles, etc.
 - Ej: desarrollo -> alfa -> beta -> lanzamiento
 - Administración de construcción...
 - Mecanismo de control de acceso...



Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Función 2: Control de la Configuración

- **Administración de construcción.** Se debe dar seguimiento de cómo el software se construye, para poderlo construir apropiadamente
- **Mecanismo de control de acceso.**
 - Permitir sólo el acceso y cambios autorizados
 - Permitir “checkout” (para propósitos de cambio) a sólo un individuo a la vez
 - Usar copias privadas funcionales para hacer modificaciones (hasta que sean probadas)



TECNOLÓGICO
DE MONTERREY.

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Función 3: Administración de Cambios

Mecanismos de autorización de propuesta de cambio

- Para saber **por qué** se está haciendo un cambio y **quién** es responsable de él
- Suele hacerse por un **Comité de Control de la Configuración** para asegurarse que las partes afectadas estén informadas
- Mantiene registros de cambios
 - **Registra lo que se hizo, por quien, y cómo probarlo**
 - Para poder **recrear** cualquier estado previo, rastrear todo cambio de la línea de referencia más reciente, etc.



Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Función 4: Contabilidad de Status

- Mantiene **descripciones e información de status** en todas las versiones del software
- **Descripción de la línea de referencia actual**
 - Todos los elementos - nombre, descripción, status (en cambio, a repararse, etc.)
 - Todos los cambios - tipo, por qué, quién, cuándo, causa del problema, etc.



TECNOLÓGICO
DE MONTERREY.

Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Contabilidad de status (continuación)

- **Descripción del lanzamiento o versión actual vs. siguiente**
 - ¿Qué cambios se incluirán?
 - Status de dichos cambios
 - Estadísticas como qué tan seguido se cambió, etc. para cada componente de la línea de referencia



Repsao CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Contabilidad de Status (continuación)

- Descripciones y copias de versiones anteriores
 - Archivando si no se usa más
 - Accessando si todavía se usa
- Registro de toda actividad de cambio
 - Registro de todo cambio desde la línea de referencia
 - Para cualquier elemento, identificar todos los cambios, usuarios, y usos para ese elemento
 - Para cualquier cambio, identificar todos los elementos afectados



Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Función 5: Autentificación de la Configuración

- Auditoría Funcional de la Configuración
 - Verifica que el software haga lo que se supone que debe hacer.
- Auditoría Física de la Configuración
 - Verifica que todas las partes esten ahí y que sean las partes correctas.



Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

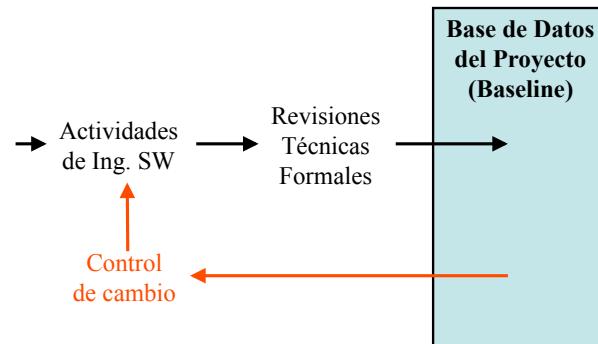
Responsabilidades de la CM

- El administrador de la configuración es el punto central para cambios del sistema y tiene las responsabilidades siguientes:
 - Desarrollar, documentar y distribuir los **procedimientos de SCM**
 - Establecer la línea de referencia del sistema (**Baseline**), incluyendo provisiones de respaldo
 - Asegurar que no se hagan **cambios No autorizados** a la línea de referencia
 - Asegurar que todos los cambios a la línea de referencia sean **probadas regresivamente**
 - Proveer un punto de referencia para la resolución de excepciones



Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

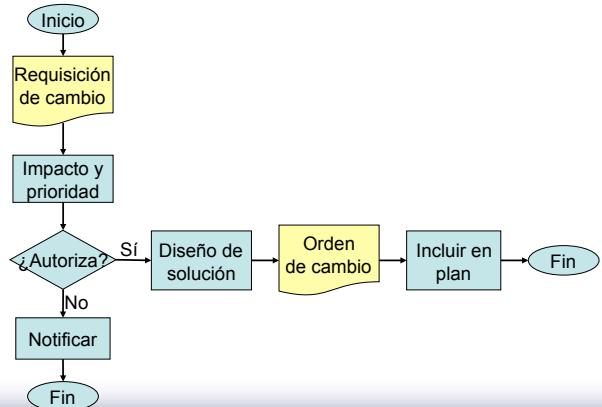
Control de Cambios



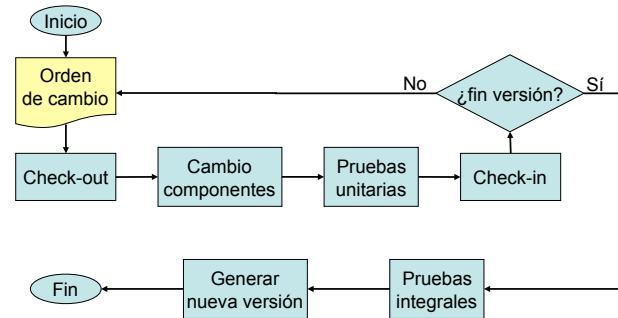
196

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Proceso de Control de Cambios

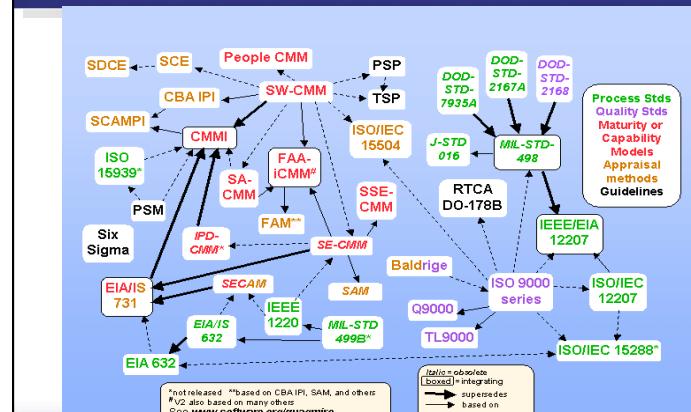


Proceso de control de versiones



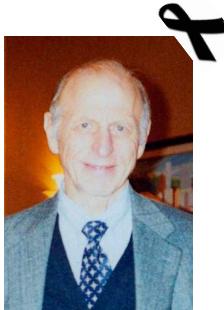
Modelos de Madurez del Software

Modelos de Calidad en el Proceso de Software



CMM - Un marco de Referencia de la Madurez

- Proyecto iniciado y dirigido por Watts Humphrey en 1984
- Se creó el Software Engineering Institute en la U. de Carnegie Mellon
- Apoyado por el Departamento de la Defensa de los EU
- Misión: The SEI advances software engineering and related disciplines to ensure the development and operation of systems with predictable and improved cost, schedule, and quality.



Capability Maturity Model

- Deficiencias en las metodologías Incapacidad para manejar el proceso de software
 - En 1986, SEI (Software Engineering Institute): marco de trabajo sobre madurez de procesos
 - En 1991, SEI desarrolló Capability Maturity Model (CMM)
 - Conjunto de prácticas recomendadas en determinadas áreas clave de proceso
 - **Mejora la capacidad del proceso de software**
 - Guía en la selección de estrategias de mejora de proceso
 - **Establecer la madurez de los procesos**
 - Determina cuestiones críticas para la calidad y la mejora del proceso
- **22 áreas de proceso.**

CMMI

En una organización **inmadura**:

- Procesos de software: improvisados o no respetados (si existen)
- Planificación en función de los problemas
- Presupuestos y planificación incumplidos
- Sin base objetiva para evaluar la calidad o para resolver problemas
- Inexistencia o reducción de las actividades de mejora de la calidad

En una organización **madura**:

- Capacidad de gestión: desarrollo de software y procesos de mantenimiento
- Proceso de software difundido al equipo y planificado
- Procesos modificables: pruebas piloto controladas y análisis de coste/beneficio
- Roles y responsabilidades establecidos en el proyecto y la organización
- Gestores: monitorización la calidad de los productos y de los procesos
- Planificaciones y presupuestos realistas: rendimientos históricos
- Proceso disciplinado en el que todos los participantes entienden su valor, existiendo además la infraestructura necesaria para soportar el proceso

CMMi: Representación ESCALONADA

NIVEL DE MADUREZ

5 OPTIMIZADO
4 GESTIONADO
CUANTITATIVAMENTE

3 DEFINIDO

2 GESTIONADO

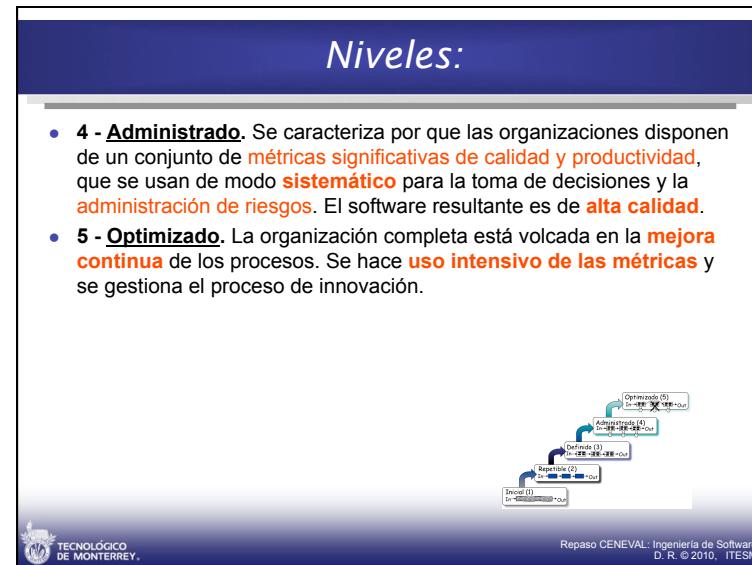
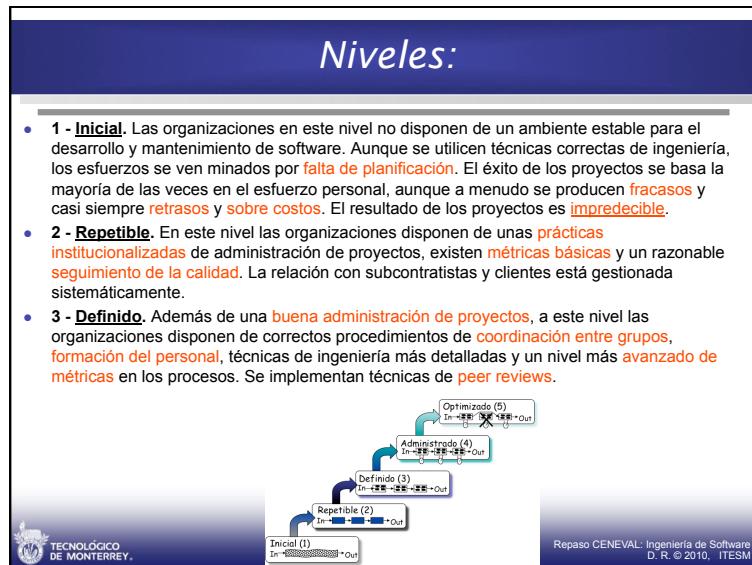
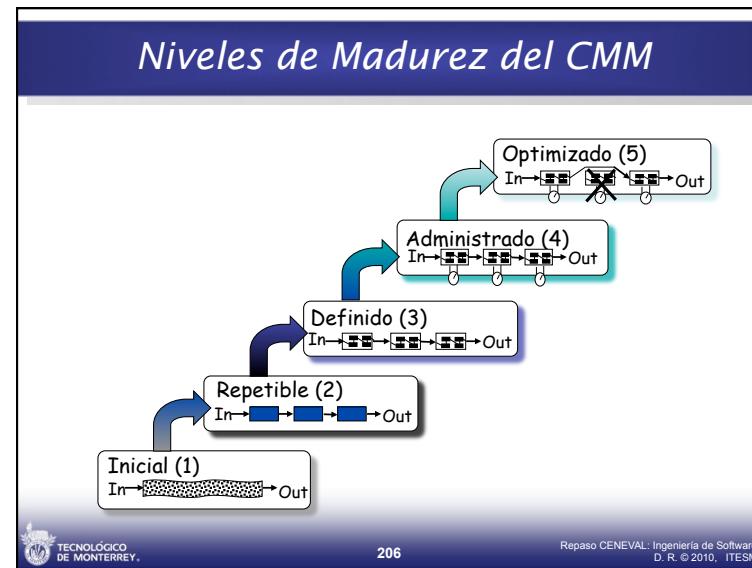
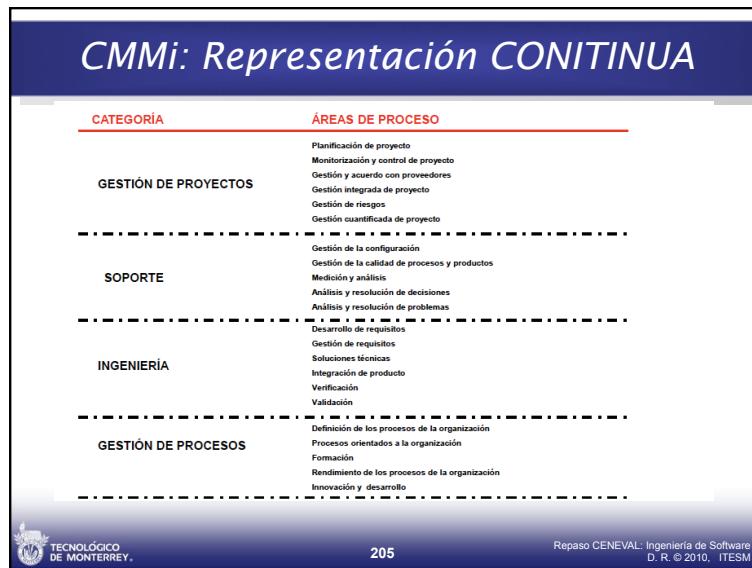
1 INICIAL

ÁREAS DE PROCESO

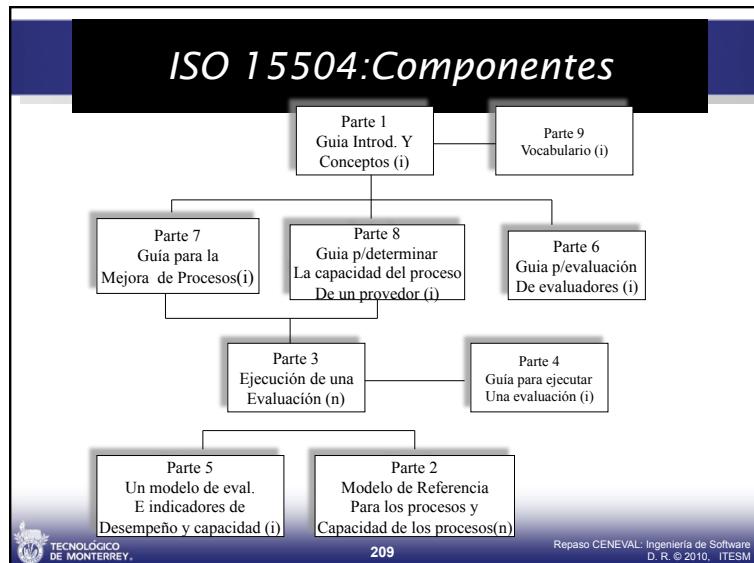
Innovación y desarrollo
Gestión cuantificada de proyectos
Rendimiento de los procesos de la organización

Desarrollo de requisitos
Solución técnica
Verificación
Validación
Integración de producto
Procesos orientados a la organización
Definición de los procesos de la organización
Formación
Gestión integrada de proyecto
Gestión de riesgos
Análisis y resolución de decisiones

Gestión de requisitos
Planeación de proyecto
Monitorización y control de proyectos
Gestión y acuerdo con suministradores
Medición y análisis
Gestión de la calidad de procesos y productos
Gestión de la configuración



ISO 15504:Componentes



Que es SIX Sigma

- Una **medición estadística** del desempeño de un proceso/producto
- Un **objetivo** que busca la perfección en la mejora de desempeño
- un **sistema de gestión** que busca el liderazgo de la organización bajo un desempeño de clase mundial

210

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Tabla de Medición de Six Sigma

Defectos por Millón De oportunidades	Nivel de cumplimiento	Nivel SixSigma
690,000	31%	1
308,537	68%	2
66,807	93%	3
6,210	99.38%	4
233	99.9767	5
3.4	99.9997	6



211

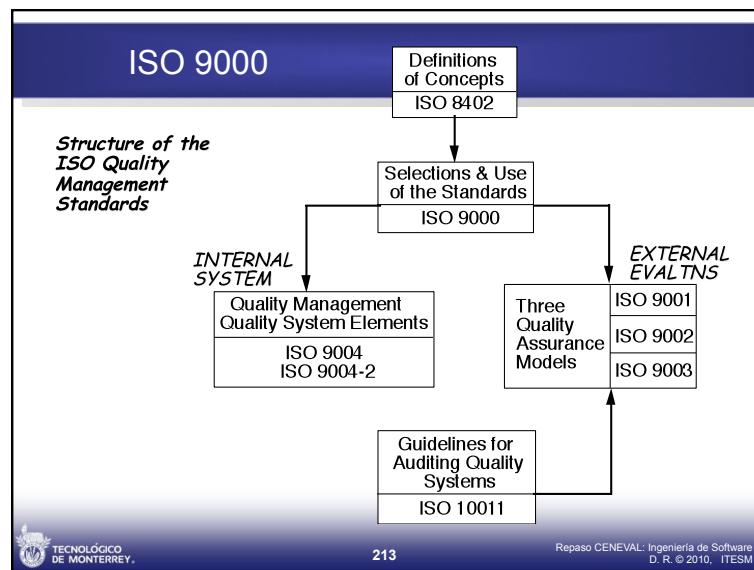
Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM

Proceso de solución de problemas: DMAIC

- Define, Measure, Analyze, Improve, Control
- Equipos de 3 a 10 personas, preferible entre 5 a 6
- Equipo diverso: áreas, competencias, habilidades, experiencia, edad..
- Equipo de Igualas

212

Repsio CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM



Dentro de ISO9000

- 9000-1 Normas para la selección y uso
- 9000-2 Normas generales para la aplicación de ISO 9001,9002,9003
- **9000-3 Normas para la aplicación del ISO 9001 al desarrollo, suministro y mantenimiento de software**



214

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM



Mucha Suerte en el CENEVAL

Ing. Rafael Salazar Chávez
Mtro. Gustavo Cervantes Ornelas
Ing. Jakeline Marcos Abed

Repsos CENEVAL: Ingeniería de Software
D. R. © 2010, ITESM